POSE ESTIMATION OF HUMAN WEARING THOBE USING DEPTH IMAGES

BY

RIDWAN JALALI

A Thesis Presented to the

DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

JANUARY, 2017

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **RIDWAN JALALI** under the direction of his thesis adviser and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.

Thesis Committee

Dr. Adel Fadhl Noor Ahmed (Adviser)

Dr. Moataz Ahmed (Member)

Dr. Lahouari Ghouti (Member)

Dr. Khalid Al-Jasser
Department Chairman

Dr. Salam A. Zummo
Dean of Graduate Studies

5/3/17
Date

Dedication

I would like to dedicate this study to the Almighty God, my beloved family and friends, and to my future wife.

ACKNOWLEDGMENTS

First, I would like to thank my family: my parents, brother and sisters for supporting me spiritually throughout writing this thesis and for their continues understanding of my absence. Also, I would like to thank the management of Simulation Systems Devision in Saudi Aramco for allowing me to use the HPC environment to do my experiments. I am also grateful to the members of my committee for their patience and support in overcoming numerous obstacles I have been facing through my research. Last but not the least, I would like to thank my friends for providing me with suggestions and support in several cases.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Nomenclature

ϵ	maximum Euclidean distance threshold for a Thobe face to be classified to a certain body-part
η	Set of random features
γ	Split threshold. If $f_\theta(I, x) > \gamma$, assign x to right node, or left otherwise
$\mathfrak{D}$	Training examples reached to certain node
$\mathfrak{D}'$	Training examples reached to children of certain node
Φ	two offset vectors (u, v) from pixel x
τ	Random decision tree
F	Random decision forest
l	Leaf node
x	Training sample. In the context of body part classification, it refers to a random pixel in a training image
θ	Random feature (Φ,γ)
c	target class
c^*	Final classification for pixel x
f	Feature response formula
I	Impurity score for selecting best split feature
LBF	Lower body faces in a base-model
N_{ex}	Subset of pixels from a training image

R Depth region of size (w,h). The dimension (w,h) is sampled at random

TBP Thobe body part

THF Thobe faces in a thobe-base-model

z Depth number at pixel x

2-colors Refer to synthesized Thobe dataset labeled with 2 colors in the lower section of the body (not including feet)

4-colors Refer to synthesized Thobe dataset labeled with 4 colors in the lower section of the body (not including feet)

8-colors Refer to synthesized Thobe dataset labeled with 8 colors in the lower section of the body (not including feet)

ρ Maximum offset size to pick γ from

d depth of a tree

$H(\mathfrak{D})$ The entropy in the class distribution of a certain node given $\mathfrak{D}$

T Number of trees in a random forest

WCB Western Cloth Classifier

Denil Refer to Denil's public synthesized dataset published in [147]

RGB-D Pair of colored and depth images

TFwT Thobe faces that lie within a threshold from a body face

THESIS ABSTRACT

NAME: Ridwan Jalali

TITLE OF STUDY: Pose Estimation of Human Wearing Thobe Using Depth Images

MAJOR FIELD: Computer Science

DATE OF DEGREE: January, 2017

As part of computer vision field, studying and analyzing human activities gain an ironic interest in the last 10 years. An intensive amount of solutions have been proposed to process, recognize, and classify human activities using evolving environment-sensing technologies. Generally, these solutions followed one of two approaches: Model-Fitting-Based or Machine-Learning-Based. Although both approaches proved to produce outstanding results, due to the fact that human activities always accompanied with endless variations, they fail to deliver as good in unexpected or accounted for scenarios. In this study, we show how wearing Thobe can cause a dramatic drop in the accuracy of the prediction of the state-of-art machine-learning-based human pose estimation algorithm developed by Microsoft Research Center. Next, we introduce a new Thobe-specific model representation

that is used to synthesize new training dataset. We show how using such representation improves the estimation in such clothing variation and open the path towards creating more intelligent representations that accounts for western and non-western clothing styles.

ملخص

تعتبر دراسة وتحليل نشاطات البشر المختلفة من المجالات الهامة في حقل الرؤية الحاسوبية، ويظهر هذا الاهتمام جليا وواضحا إذا ما تتبعنا تزايد البحوث المقدمة ضمن هذا الإطار خلال السنوات العشرة الفائتة لاسيما مع تطور تقنيات الاستشعار في الآونة الأخيرة . تنقسم مناهج الأبحاث التي تحاول أن تفهم طبيعة حركات البشر ومن ثم نمذجتها وتصنيفها إلى قسمين رئيسين: قسم ينتهج أسلوبا يدعى "مطابقة النموذج" وقسم يستخدم تقنيات وخوارزميات الذكاء الاصطناعي. ورغم أن كلا المنهجين قد قدما لمنظومة البحث العلمي حلولا ذات نتائج رائعة، إلا أن جل هذه الحلول مقيد بمجال ما، يتسع أو يضيق كل حسب حالته. وعليه لا يوجد حل يغطي كل المتغيرات والفوضى المصاحبة لحركات البشر. في هذا البحث، نثبت هذا المعنى، إذ نظهر كيف أن ارتداء الثوب العربي يشكل معضلة لإحدى عيون أبحاث الذكاء الاصطناعي في مجال "تقدير وقفة البشر" والذي نشرته شركة مايكروسوف في عام ٢٠١١، حيث تضعف كفاءة النظام المبني من الخوارزمية المقترحة بشكل كبير إذا ما اختبرت بمثل هذا المدخل (الثوب). ثم نمضي في البحث لنقدم نموذجا ثويا ثلاثي الأبعاد استطعنا من خلاله أن نولف مجموعة جديدة من الصور المصطنعة التي استخدمناها لتدريب نظام جديد بذات الخوارزمية التي اقترحتها مايكروسوفت ولكن بكفاءة أعلى في حالة الثوب. يفتح هذا البحث الباب لمجموعة أبحاث مستقبلية توظف نماذج أذكى وأبرع للكائن البشري تراعي في هيئتها جميع متغيرات اللباس

CHAPTER 1

INTRODUCTION

Not 30 years ago!!, but since the earliest known examples of Homo sapiens more than 40,000 years ago, visualization was a method for communicating and exchanging speaking messages [1]. The vast amount of known images, paintings, and sculpturing that human has created throughout the history shows the passion he has for such creative communicating tools. Visual media allows for wide meanings and feelings to be communicated and explained in short representations.

Since the beginning of the modern scientific world, many studies have been proposed to enhance the creation, modification, and interpretation of visual media. This has led to the introduction of computer visualization concept that underlies all the astonishing visual worlds that took us from blurry paintings inside dark caves to the countless number of 3D movies, games, and simulators. Recently, more thrilling concept "Computer Vision" has been introduced. Basically, it is to provide computer with an artificial eye (e.g. Camera) that allows it to process, analyze and understand the surrounding environment. In this research, we are

concerned with human pose estimation. Pose estimation is the problem of determining the 3D transformation of an object from its projected view on a 2D frame. Similarly, human pose estimation is the problem of identifying the transformation of a human standing or moving in the range of the sensing device. The pose of a human is usually estimated by inferring or identifying the 3D positions of human joints.

1.1 Background

In order to estimate human pose, 3D information must be represented in primitive form. Several techniques have been proposed to obtain and record 3D information of the scene and thus give the machine a primitive data that can be processed [2]. One such technique is to use real-time pattern projection and triangulation. Microsoft Kinect is the most well-known device example in this family. With the announcement of Kinect, regular user with a relatively cheap price could own a 3D motion sensor that can be utilized to develop interactive applications.

Recently after the release of Kinect, people started proposing methods and algorithms to exploit and evaluate Kinect ability to detect human body and all possible physical poses (e.g. [3] [4] [5]). In 2011, Microsoft revealed the algorithm used in their implementation of Kinect human pose recognition system [6] Microsoft research team employed a random forest classifier to train the system to identify body parts location and then propose a 3D positions for the joints from depth images only . The team used synthesized depth and labeled images

by recording 3D human base-models. In [6], the team claims that their system would estimate body parts invariant to pose, body shape, clothing, etc. However, unlike western jeans or sport pants, in many cultures, people wear a single piece of fabric that covers the entire body. In such cases, the lower body would look like a concrete opaque square in a 2D image hiding all the spatial details of the left and right thighs, knees, and legs. In this research, we evaluate the performance of the present Kinect's human skeletonization system on cases where the body frame will include people wearing non-western clothes. Specifically, we use Thobe, an ankle-length garment, usually with long sleeves, similar to a robe. We show that in such case, Kinect joint detection framework fells short of achieving acceptable estimation. Then we propose a Thobe-specific base-model that we use to synthesize Thobe-specific training dataset. Unlike [6], where textured mapping was used for labeling, we propose a dynamic labeling technique to account for the chaotic behavior of the Thobe. Following the same training and prediction procedure in [6], we show that random decision trees trained with Thobe-specific dataset can achieve good estimation of the body parts of a human wearing Thobe.

There are many applications that can benefit from this research and its future extension. In gaming industry for example, employing dress specific models can extend the dressing freedom of the players and extend the scope of real-environment games requirements (e.g. Martial arts). Also, in fashion realm, systems can be trained to track the smoothness of the legs movements with particular size of the Thobe or similar dresses.

1.1.1 Computer Vision

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. Similar to human eye and brain, the paradigm in computer vision is to provide a machine empowered with processing and analyzing mechanisms (brain) with primitive data that describes the surrounding environment through a sensor (eye) [7].

1.1.2 Vision Devices

In this section, we briefly discuss three type of sensors' technologies that have been used to generate the row data for the various computer vision applications: image sensor, range sensor, and stereo sensors. Again, the main task of these sensors is to provide information about the scene to be analyzed in some digital representation.

Image Sensors

An image sensor is a sensor that provide information about the scene in a form of image. The common term for such sensor is Camera. All modern cameras, whether recording static frames (images) or continues (videos), generate colored view of the scene. Colors provide powerful information for computer vision analysis. As Gegenfurtner and Rieger has stated in their work for measuring the role of colors in visual processing: "color helps us to recognize things faster and to re-

member them better [8]". Colored cameras are highly available, and hence colored images are captured everywhere by everyone. The recent trend in social media has shifted toward visual communication via colored images and videos. Millions of selfies are taken everyday by people around the globe. Unless this changes, computer vision researchers, wether directly or by converting them to grayscale, have to deal with colored images. Tanaka and Hagen discuss the role of color in achieving high level vision in [9, 10] respectively.

Range Sensors

Unlike traditional cameras, range sensors provide a metric information of the distance between the camera frame and the nearest objects. Popular example of range sensors are *Time of Flight sensors* (called LIDAR sensors). As its name indicates, the principle is to shoot a photon at the environment and calculate the time it takes to reflect back. This sensor or camera is used widely in machine vision, robotics and ocean and earth topography [11]. *Structured light* [12] is also a depth sensor. The mechanism it follows, however, is slightly different then ToF cameras. Structured light systems project a known pattern of light onto a scene, and the depth is reconstructed from the distortion of the scene by the objects. There are many applications for structured light cameras. Google launched it's first mobile with 3D sensor named *Tango* which includes Structured light [13]. Also, Intel employs structured light sensor in its Intel *RealSense* gesture and virtual reality camera [14]. Microsoft Kinect camera uses is based on structured light. We provide extended detail about ToF cameras, Kinect and depth images

in Section 2.1.

Stereo Sensors

Stereo sensors use two adjacent, identical, cameras to capture the scene. This allows the camera to capture 3D images. Moreover, stereo cameras can be used to capture depth information. Since the camera is empowered with two views, measuring the depth can be done by computing corresponding points between two views of a scene and then calculate their depth using triangulation. Stereo cameras are widely used in autonomous vehicles [15]. Moreover, the depth information generated from stereo cameras are used in computer vision applications like hand gesture control [16] and robot navigation [7].

1.1.3 Vision Tasks and Applications

The theories and principles of computer vision spans a wide range of useful applications. Among these are assembly and material handling, automatic target recognition, photo interpretation, navigation, automatic inspection, digital medical diagnosis, interactive gaming and extraction of three-dimensional structure. These applications operate by executing one or multiple computer vision tasks. The function of the computer vision research society has been, always, to create solutions for these tasks and continuously work on improving them and provide alternatives. Below, we briefly describe three of the most essential computer vision tasks: image processing and segmentation, recognition, and motion analysis.

Image Processing and Segmentation

While not direct application of computer vision, image processing algorithms and techniques are always associated with computer vision tasks. The setup of the recording or for the sensing data to be analyzed might not be suitable for further analysis. Therefore, image processing tools help to convert the data into a form that suffice the prerequisites of the algorithms. Examples of such tools include 'exposure correction and color balancing, the reduction of image noise, increasing sharpness, or straightening the image by rotating it. More connected to computer vision than image processing is image segmentation. In this task, the goal is segment the image into groups of pixels that semantically fit together. This task has been widely studied and hundreds of algorithms have been proposed [17]. Figure 1.7 (at the end of the chapter) shows examples of different segmentation techniques. Segmentation techniques allows to extract the object of interest from the image for further computer vision analysis.

Recognition

The classic and the most challenging problem in computer vision is to analyze and recognize all the constituent objects in the scene [17]. Thousands of technical papers have been published and several competitions have been conducted in the past decade to improve machine recognition, and yet due to the extreme variations in the shape and appearance of objects, no work achieve optimality [18]. Currently, the recognition problem is broken down to several sub-tasks that can be handled separately. If there is a well defined object to be recognized in images or videos, then the problem is *object detection*. Face recognition in cameras and

pedestrians detection systems in smart cars are perfect examples of applications benefited from object detection solutions. Another task in recognition is *object classification*. This involves training a system to classify multiple objects into classes. A common technique to tackle this problem is by identifying distinguishing features for each object. These features are, then, searched for in the image or the video. Practically, object classification solutions set the ground for marvelous applications. Among those are automatic video content annotation [19] and skin cancer screening [20]. Content-based image retrieval or image search is another problem in recognition domain. Given an image, find all images or content that relates to this image. The search algorithm in this problem can understand the content of the image by matching it with similar annotated images. The difficulty of the problem is to create smart definitions and algorithms to measure the similarity between images. Google now allows to search its content using images: a feature that can be used to retrieve pictures and information about unknown people or unknown places.

Motion Analysis

Motion analysis is part of computer vision studies that is mainly concerned with analyzing the movements and the deformation of objects from one or multiple images. Several sub-tasks fit under motion analysis: object detection and tracking, object pose estimation, activity and action recognition, egomotion, and optical flow [21]. This area of computer vision has received an increasing attention in the past 10 years. The interest comes from the fact that the research associated

with it are highly used in entertainment and security application such as athletic performance analysis, interactive gaming, augmented reality, surveillance, monitoring of traffic, monitoring people and their activities in public places and many others.

In this master thesis, we work on a problem that is related to motion analysis. Specifically, human pose estimation with covering dress. Human pose estimation is a key problem in computer vision. Its importance is drawn from its current and future applications. For example, avatar animation with marker-less motion capture technologies is based on pose estimation techniques. Interactive gaming and virtual reality applications are another example that is highly depended on pose estimation technologies. In the future, autonomous cars and driver assisting systems must be empowered with human pose estimation tools to understand the behavior of pedestrians. Also, robots must understand human activity to allow for intelligent decisions and interactions. In the following section, we shed lights on the works are closely related to ours.

1.2 Related Work

Human pose estimation is closely related to other computer vision tasks: mainly, human motion analysis, gesture recognition, human action recognition, and human motion tracking. Figure 1.1 shows the related tasks and short description of each. In the following sub-sections, we briefly discuss two tasks related to human pose estimation: human detection and tracking, and human action recognition. The

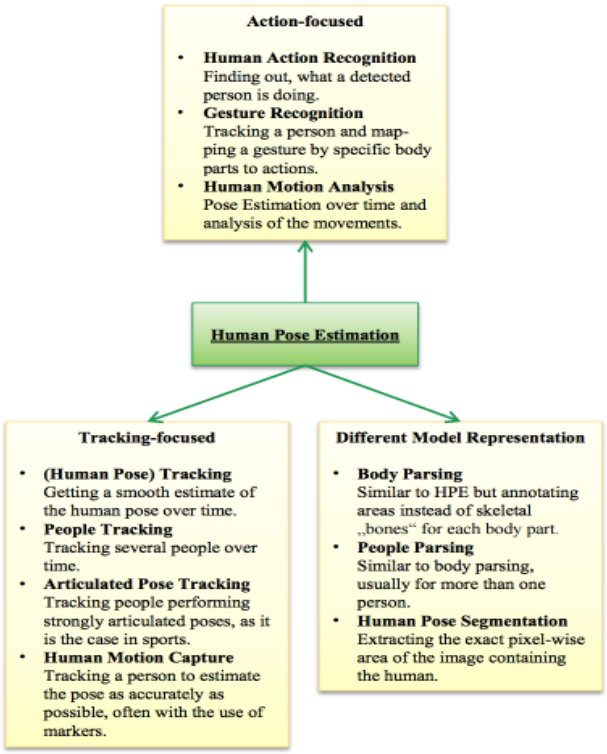


Figure 1.1: Related tasks to Human Pose Estimation [22]

former is usually a prerequisite for pose estimation and the latter is an application that capitalizes on pose estimation. Next, we present an overview and critical analysis for human pose estimation literature.

1.2.1 Human Detection and Tracking

Sensing people in a sensor environment is a fundamental problem in computer vision. Perhaps all the tasks related to human-machine-interaction rely explicitly or implicitly on some human detection and parsing solution. In surveillance sensors, for example, regions of moving objects are first segmented from the rest of the image before subsequent processes follow [24]. Figure 1.2 shows an example of architecture of on-board pedestrian detection and tracking system.

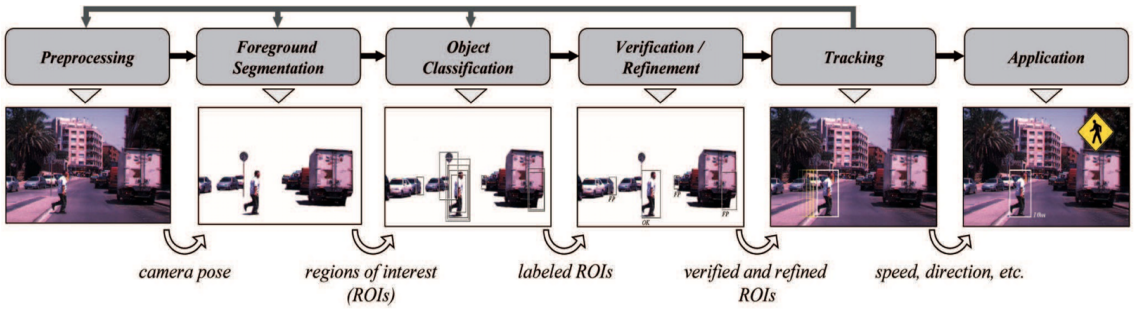


Figure 1.2: An example of on-board pedestrian detection system[23]

The first step in an active computer vision system, is to detect an object movement. Upon the discovery of such disturbance in the scene, the pixels around the object is identified for further analysis. In [25], Perekh discusses three types of detecting moving objects that they have compiled from the literature: frame differencing, optical flow, and background subtraction. Assuming a moving object is detected, the next step would be to detect whether the object is a human or not. Human detection has been widely studied in the first decade of this century. Ogale provide an overview of multiple papers published before 2006 [26]. He finished his article with an overview of one of the most referenced work in computer vision by Dalal and Triggs [27]. In 2005, Dalal and Triggs introduced a human detector based on Histogram of Oriented Gradient (HOG) feature. According to Dollar, nearly all modern detectors uses HOG feature or a variant of it and no single feature could outperform it [28]. In Massimo survey of human tracking in RGB-D [29], we can count 11 out of 17 works that use HOG representation with RGB images. Dalal and Triggs detect human in an image by representing each example with a single, fixed HOG template and then learn an SVM classifier that decides whether a human exists in an image or not. Another renowned work, which

preceded the work of Dalal and Triggs, is the detector of Viola and Jones that uses Haar wavelets and AdaBoost classifier for face detection [30] and later for pedestrian detection [31]. Another line of technique for human detection is to use shape features. To describe the shape of a human, edge-based features (e.g. shape-edge [32], edgelet [33], and shapelet [34]) are usually employed [35, 28] Figure 1.3. The basic detection technique with shape feature is to compare the input example to a predefined shape templates using space metrics like Hausdorff [32] or Chamfer [36]. Wu and Nevatia used AdaBoosting classifiers similar to Viola and Jones' to learn body part detectors that are combined with a joint Baysian likelihood function to detect possibly occluded humans. Finally, motion features are used in some literature to discriminate human from another moving objects. Similar to what we mentioned above, the motion of the human can be described using temporal differences or optical flows. Viola, for example, uses a temporal extension of the Haar features in [30] to encode the temporal difference in the motion of pedestrians. The combination of appearance and motion information resulted in large performance gain. A comprehensive overview about human detection in general and in smart-car systems be found in [28, 25, 35, 26, 37, 23, 38]

After a human is detected in an image or a frame of a video or an active recording, a tracking system can operate on next images. The main objective of trackers is to reduce the search space of detected people by using lighter trajectory representations. Nevertheless, some real-time detection algorithms can act as detectors and trackers. Trackers often represents their tracking target region

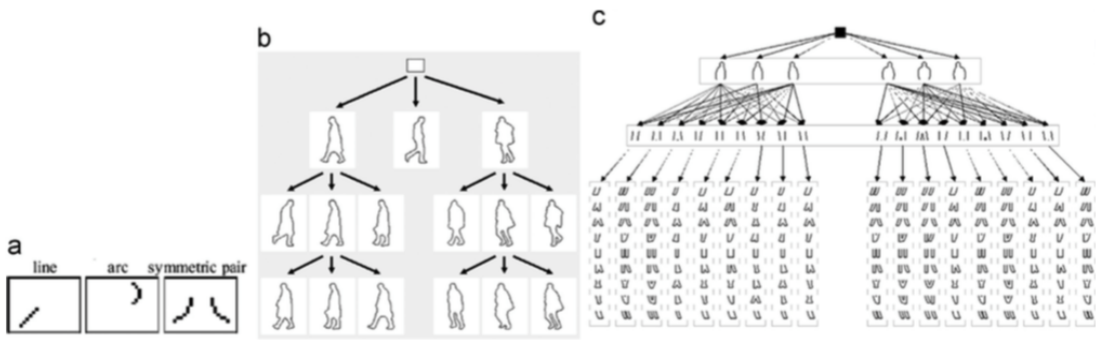


Figure 1.3: Examples of edge-based templates from [35]: a) edgelet, b) full body shape, c) part templates

using bounding box [39]. This representation, while might include noisy background pixels, reduces the number of parameters [40]. Trackers might also use contour representation. Such representation gives higher freedom for the change in the target appearance. It is used in [41] to track human body, human face and objects (e.g. soccer ball) with moving cameras. However, such representation only good for well defined target objects and cannot be easily extended for general purposes. Figure 1.4 shows other types of trackers' target representation. In order to detect human, or any other object, in succeeding images, the appearance of the detected human must also be encoded in some representation. The selection of the representation depends heavily on the scene. For static scenes, one can use the advantage of the static background and employ background intensity representation [42]. The target is represented in this case by its probability density function (pdf). For non-static scenes, which is almost always the case, the target can be represented by array of brightness values [40], color histogram [43], or feature vectors [44]. The selection of these representation depends on the nature of the target and the tracking method. For example, the array of bright-

ness is disadvantaged when an albedo changes is frequently seen. Given a current target region and the target representation, the detectors' function is to locate the best region of the target in the next frame. Candidate regions can be either sampled [45], searched [46], or predicted (e.g. using Kaman filter [47] or particle filter [48]). The basic approach to investigate the candidates is to find the best match. This is usually done using Mean-Shift method. The tracking method tries to find the area that is most similar to the template by maximizing a similarity score [49]. This approach relies heavily on the similarity of the target appearance and thus prone to degraded performance when sudden appearance changes occur. Another approach is to detect using discriminate classification. Such approach seeks to avoid selecting background areas by using online or dynamically trained classifiers from the so-far seen frames [50, 43]. Unlike template matching, this approach capitalizes on the advancements of classification methods and therefore less sensitive to sudden changes in the appearance. However, the trackers would depend heavily on dynamic classifiers which can confuses the detection in case of wrongly labeled data. Extended survey about human tracking advancements can be found in [39, 29, 25].

1.2.2 Human Action Recognition

One of the interesting task in computer vision is enabling machines to understand human actions and interact with them. This includes recognizing what a person is doing, or predicting what he/she is planning to do. Also, many researches

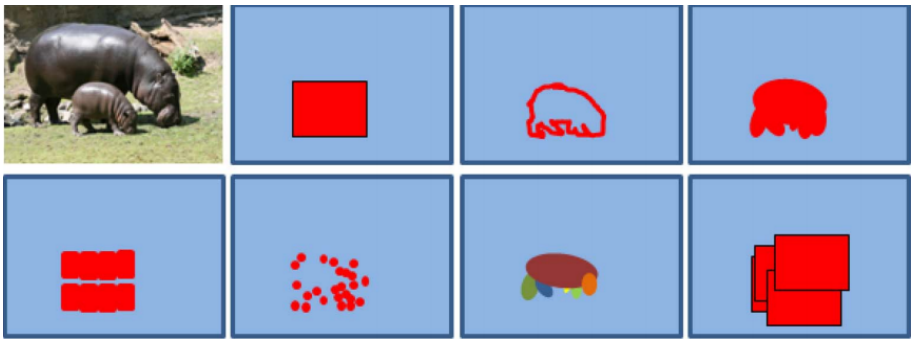


Figure 1.4: Target region representations used in tracking from [39]. From left to right, top to bottom: target bounding box, target contour, target blob, patch-based, sparse set of salient features, parts, and multiple bounding boxes

focused on human gestures and proposed algorithms to computationally interpret them. The surveys in [51, 52, 53, 54, 55, 56, 57, 58] provide a comprehensive overview of the works have been done in this area. As far as human action is concerned, many approaches have been employed to tackle the problem. There are two ways in which approaches differ: human body representation, and the action classification technique used for recognition

In terms of human body representation, the methods proposed boil down to two main domains: methods that employ human body models and methods that don't. In the first, the skeleton pose, or any other human body representation, is estimated in each frame and the action or the activity is recognized from the joined clip of estimates. Human model representation includes but not limited to: silhouettes [59], contours [60], Histogram of Oriented Rectangles (HOR) [61], and trajectories [62]. The model-based representation is powerful since it encodes a very important details about human. However, they depend heavily on pose estimation, and background subtraction. Non-model based representation describes an event or an action as a collection if independent patches. Non-model based

methods tackle the problem by identifying local features. The features are interest points that are detected from the motion [63] or from the dense [64].

When full human body, hand or head is identified and represented in a sequence of frames, the problem is converted into classification problem. Certain action, nod, or gesture meaning is assigned for each sequence of frames. The classification is accomplished using different statistical machine learning techniques with three main difference: methods that implicitly model or pays not attention to temporal variations, methods that model temporal variations, and methods that use unsupervised classification or no action modeling. In the first, the action is modeled with a single representation or per-frame. A leading work is presented by Bobick and Davis in [59] where they introduced temporal templates through projecting frames onto a single image. In this approach, the recognition is carried out using nearest neighbor classification [65] or discriminative classification (e.g. SVM) [66]. In the second, the temporal relationship is captured and the observations are considered in the sequence they appear in. The input sequence is compared with actions templates using Dynamic Time Warping (DWT) [67] distance, or state-based approach like Hidden Markov Model [68]. Finally, unsupervised learning is meant to model the global motion patterns[69]. For example, trajectory clustering is employed to learn motion pattern from surveillance subjects in live video in [70] and unconstrained videos in [71]. Unsupervised learning is meant to cover the shortcoming of Supervised approaches where only short-term local body movements are modeled.

1.2.3 Human Pose Estimation

Many methods and algorithms have been proposed to solve human pose estimation problem. Moeslund illustrates in his 2006 survey[72] more than 350 papers in this domain. Nikolaos[73], Chen[74] and Xavier[75] extends this effort to cover the works after 2008. Mainly, the notion of human pose estimation involves two process: pre-processing, and pose estimation[72]. This section discusses some of the works that have been done in each of these processes.

Pre-Processing

Background subtraction

Background subtraction or foreground detection is basically the process of eliminating all out-of-interest (background) details or pixels from the image. The remaining parts (foreground) is then carried out for further processing. Extracting the moving objects (e.g. human) in a sequence of images is a very important step in pose estimation. Both static [76, 77] and dynamic [78, 79] background subtraction have been given much attention in the past years. With the introduction of depth cameras, a modern set of background subtraction solutions have been proposed[80, 81, 82]. The simplest technique in background subtraction on static background is to subtract the first frame from every other frame and eliminate all pixels that do not exceed certain threshold [83]. Background subtraction is analogs to human detection and tracking, which we discussed in Section 1.2.1. Moreover, the work in [84, 82, 85] provide a comprehensive summary of all the

studied techniques.

Human Body Modeling

Many computer vision algorithms center on the fact that human actions can be perceived by spotting the joints spatial information. Regardless what method is used to identify the 3D location of the joints, one need first to draw the human body in some representative model. Several representations of human body model have been developed along with different estimation techniques. One common approach for modeling human body is to divide the body into rigid parts. In [86], the model of the body is represented as a skeleton with fixed width attached cylinders. In [87] the body is described as rectangular or trapezoid-shaped patches. The human body in [88] is modeled as a collection of 15 rigid parts represented via a closed triangle mesh that is constrained with kinematics chain. Xia in [5] generates a simple hemisphere to represents the head(the only part needs to be detected in their algorithm). For tracking the lower limb, [89] uses a tapered cylinders for the thigh, calf, and foot for each leg.

The main disadvantage of this approach is that it can be easily tricked if there exists objects in the image that look similar to any of the body parts [90, 91]. A more strict approach employs combined modeling of the human body. In other words, it parses the body by fitting it against a hierarchal structure of sub-models. An example of this approach is presented by [92] who employ a multi-layer model that a whole body model is placed at the top, followed by a combined parts and finally rigid body parts.

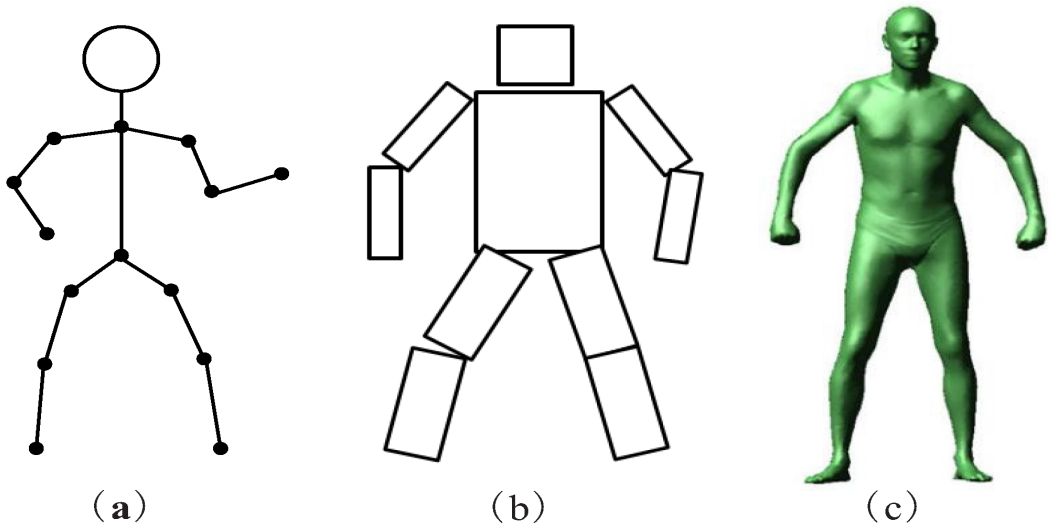


Figure 1.5: Examples of human model (a) Skeleton-based model; (b) rectangular batches model (c) realistic 3D human model [95]

The previous models imagine an abstracted view of the human body in order to overcome the complexity associated with different body sizes and lengths. Another research path accepted the challenge associated with this complexity and used models that mimic human look and take into consideration the mathematical constraints associated with it. Shotton in [6] uses a 3D models of 15 varied base characters, both male and female, from child to adult, short to tall, and thin to fat to build his machine learning dataset. The human body is modeled as a set of 31 body parts, each encoded with distinct color. Bigontina in [93] reduces the human descriptor to 14 body parts that are to be identified from silhouettes representation to predict the pose of soccer players in broadcast images. Figure 1.5 shows examples of human body models described above. More examples and further description are presented in [94, 73, 95]

All the previous models represent the human body with forked lower section. In other words, they assume that there is a always a free space between the two

legs that is distinguishable with any sensing device. However, in this research, we consider a case where the character is wearing a cloth that covers the space between the legs and thus violate the assumption. To overcome this issue, we use 3D base models of a man character wearing Thobe to represent the character (Figure 2.1). The human pose is then estimated by segmented body parts similar to [6, 93] with an extra Thobe Body Part (TBP). This model is discussed in details in Chapter 3

Pose Estimation

Pose estimation surveys provide different taxonomies and classifications for existing pose estimation methods [74, 96, 94, 97]. While outdated, the general classification used in [94] is still valid and is adapted by the recently published 3D human pose estimation survey in [73]. Poppe divides the estimation techniques into two main categories: Top-down approaches (generative: match a projection of the human body with the image observation), Bottom-up approaches (discriminative: assemble a human body from identified body parts), and hybrid approaches.

The basic and the common approach is to fit an artificial body model into the foreground points [98]. In [86], human pose can be estimated by searching a scope of poses and choose the one that optimizes the likelihood function that compares synthesized depth images of the model to the observed image. Another probabilistic model is presented in [88]. In this approach, the pose is estimated with a Bayesian network that put under consideration the conditional probabilities of the random accelerations, configuration of the body and consequently the depth

measurement over the time. Similarly, [89] adopts a constant velocity model that is distanced from the observed depth image returned by a Kinect camera to estimate the lower body state over time. A basic square error function is used in [5] to fit a detected circle in the observed image to a 3D head model. Agarwal [99] maps poses directly from silhouettes. The main advantage of generative methods presented above is their ability to estimate the pose with relatively higher accuracy. However, they require the estimated pose to be present in the pose space. Moreover, whether the search algorithm considers previous frames configuration, or merely brute-force, it is overall computationally expensive due to the unlimited number of possible poses.

On the other hand, a Bottom-up methods or discriminative approaches estimate the human posture by assembling body parts from a predefined 2D template. Two main features in this approach. First, the estimation algorithm can take into consideration physical constraints [94]. For example, one can detect the torso point, and use that point as a validation reference for the detection of other parts. Second, a per frame initialization can be used. So, if a detection is failed in a certain frame, it could be fixed in the next frame without error accumulation. Discriminative approaches have the advantage in execution time because the employed models have fewer dimensions [73]. Plagemann in [100], proposed an approach to estimate the location and orientation of three body parts: head, hand and foot based on geodesic extrema on a 3D mesh. They find the point of interest by tracing the direction of the extremity. This approach is limited as it

only identifies three parts and doesn't distinguish left from right. Shotton employs random forest to achieve real-time estimation[6]. The problem is converted first into multi-class recognition by introducing an intermediate representation of the human body. The intermediate representation is then used to vote for the joints location. In [101], Girshick, building on hough forest, shows that using regression from the raw depth image instead of classification of an intermediate representation improve the estimation and outperform the estimation of Shotton. Both algorithms exploit no temporal information, allowing them to re-initialize in each frame and be free from accumulating failure. This is an advantage when accurate estimation of all frames are compromised by continues good estimation at high speed. However, this becomes disadvantage when each frame is considered to be important as none of the two approaches provide adjusting or optimizing mechanism for inaccurate estimation.

Hybrid approaches combines generative and discriminative approaches to improve the estimation and the computation time. In [102], Holt proposes a hybrid approach of part-based and example-based estimation. Holt applies a modified Random Decision Forest to identify Poselet (set of parts that are tightly clustered in configuration space and appearance space) activations. Each Poselet is trained to define a set of key-point predictions. The method then infer joints by combining key-point predictions from Poselet activations within a graphical model. Because it is computationally expensive, this approach is meant for static images and cannot be used for real-time estimation. Moreover, Holt's system, as it is, can

be used for detecting the pose of the upper body only. Extending the approach to handle the lower section requires creating a new set of Poslets. Shum proposes in [103] a reliability measure to evaluate the correctness of the estimation achieved by Kinect camera pose estimation framework [6]. Further optimization of the estimation is then carried out by querying the observed pose with the reliability rate into a database of postures to find a more reliable estimations. The problem with such method, is that it compromises the speed of the recognition algorithm. Moreover, the method assumes that a similar posture to the exposed posture exists in the database which is not always true[100]. More works that follows this approach can be found in [94, 73]

1.3 Pose Estimation with Covering Dresses

All the mentioned algorithms in the previous section are meant to operate on images where the character exposed is wearing western-style clothe or, in general term, has fork-shape lower section. However, by reading the associated papers, and looking at the training and the testing datasets, one can hypothesize that they will not operate on a covering garments and dresses that are worn in a large area of the world: Middle East and East Asia. Skirt, Japanese Kimmo, Indian Sari and Arabic Thobe are examples of such dresses Figure 1.6.

As mentioned earlier, Shotton proposed a real-time solution for human pose estimation from depth images only. However, as will show in Chapter 4, it fells short when the character is wearing Thobe. Thobe is a covering dress that the

Figure 1.6: Examples of covering dresses

lower section of the body is hidden inside the draped floating fabric and only the feet are appearing. In this research we capitalize on Shotton approach, and introduce Thobe-specific framework to improve estimation of the pose. Specifically, we concentrate on the lower section of the body and redefine the body segmentation proposed by Shotton in a way that addresses the added complexity of the Thobe.

Figure 1.7: Some popular image segmentation techniques from [17]. (a) active contours; (b) level sets; (c) graph-based merging; (d) mean shift; (e) texture and intervening contour- based normalized cuts; (f) binary MRF solved using graph cuts.

CHAPTER 2

FOUNDATION AND RESEARCH APPROACH

This master thesis focuses on evaluating and improving the current state-of-art human pose estimation algorithm [6] (we are going to call it Shotton from now on) with unforeseen clothing variation.

2.1 Dependent Technologies

2.1.1 Time-of-Flight Camera

Time of Flight Camera (ToF) is one type of sensor devices that generally produces primitive data for the surrounding environment for computer analysis. The two major components of any ToF cameras are optical transmitter and optical receiver [104, 105]. ToF cameras provides a pixel-wise distance or range for the 3D view by projecting light to the scene and observing the reflected light. The modern devices

or applications of this principle gains more advantage by using multiple laser beam to illuminate the scene. This pixel-wise parallel measurement therefore provides high-frame rate that is a vital requirement of many computer vision research, technologies, and applications. As of this, the distance is measure for every pixel, and a depth map is constructed where each point in the map corresponds to a 3D point in the scene (voxel) (e.g. Figure 2.1). A 2D representation of this map, a usual form of the input of any computer vision application, can be resolved by a gray-scale image such that the further the voxel, the darker the intensity.

Figure 2.1: Depth map of a soda cans [105]

Among other sensors, ToF cameras provides a sensory data that is, if not fully primerly, unaffected by the color intensity of the scene. This feature is crucial to many object recognition problems solvers. Moreover, lightning, shadows and shading factors can obscure the objects and hide many of its imperative details.

2.1.2 Microsoft Kinect

Computer vision applications and software research rely primarily on sensors hardware. In the past, few computer vision researchers have had access to or were able to afford a reliable range sensor device. It was only on 2010 when microsoft launched Kinect, that researchers were able to test and qualify their computer

vision algorithms on reasonable-price, sound-quality and easy-to-acquire sensing device. Microsoft continues to develop and enhance Kinect since then. A very small google search on the scholar domain reveals 111 technical papers with the word "Kinect" in the tile on the first 3 months of 2015.

Kinect sensor captures both depth and color frames at a 30 fps rate. The sensor contains and infrared laser emitter, and infrared camera, and RGB camera. Unlike common ToF cameras, Kinect uses a modified version of the send-and-receive approach to calculate the depth map which is called structured light. The basic idea behind structured light is to project a light pattern into the 3D scene, and construct the depth map from the distortion of that pattern. The pattern is emitted as speckles or spots from the IR projector while the CMOS IR camera observes it. The depth data is calculated from the triangulation of each speckle between a known calibrated pattern and the shifted observed pattern. The calibration between the projector and the camera is configured at the manufacturing time (full description can be found in [106])

2.1.3 Depth Images

It has been long since visual data and 3D world numerical representation was available. In order to analyze and compute static or dynamic description of the world, a numerical or symbolical interpretation procedure must be selected. In fact, there is a whole area of research called *data fusion* that aims to provide methods for combining several sensory data sources [107]. As far as human pose

estimation is concerned, the creation and the development of depth sensors and Time-Of-Flight cameras has had a great influence in brining cheep and in-hand devices for the research community. Depth data, images, or map contains information relating to the distance of the surfaces of scene objects from a viewpoint. Depth information has been widely used in human motion and behaviors recognition [108, 109, 110]. The main feature of depth information is that it provides a metric representation that is invariant to the color and the light of the environment [108]. As discussed above, there are several ways of how depth information can be obtained from 3D world[110]. Moreover, to obtain depth information in a synthetic rendered scene, several techniques could be used; one of which is by calculating the Euclidean distance using the well-known equation

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + (q_3 - p_3)^2} \tag{2.1}$$

2.2 Random Decision Tree

Random decision tree is a simple decision model that depicts a hierarchal decisions and their consequences. At the very end of a decision tree, a scaler value (regression) or a category likelihood (classification) is stored and can be assigned to the desired sample or instance at the end of the decision stream at prediction time.

Starting at the root of each tree, x traverses the tree downward based on its

response to the evaluation function stored at the node. Each node can split into two (binary) (Figure 2.2 A) or multiple children nodes (Figure 2.2 B). In case of binary split, the split function tests against a binary expression (e.g. true or false). Depending on the implementation, x traverses to left or right accordingly. Although binary split are dominant in decision tree literature, successive children and grand children nodes still can be detached from each other and reattached to a common grandparent (e.g. [111]). Last but not least, the output of a decision tree is stored in its leaf nodes $L = \{l_{1..n}\}$. If x reaches to a leaf node l_i, the value, either scalar or a classification probability distribution, is obtained from the node.

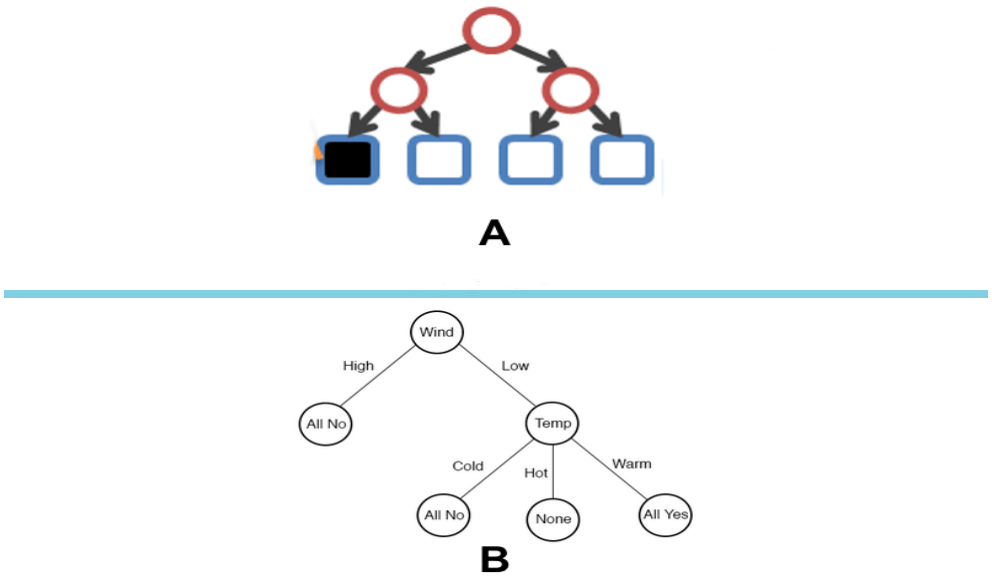


Figure 2.2: A. Cartoon showing an example of a binary decision tree[112]. Each circle depicts a node in the tree, and each square depicts a leaf node where a classification target or a regression value is stored. C_{N_i} depicts a combination function that averages the votes of the trees. B. shows an example of a multi-split decision tree

2.2.1 Training a Decision Tree

The main idea behind decision tree learning is to be able to construct a directed network of interrogative hobs that lead to one of several pre-defined knowledge. By answering the question stored in each hob or vertex in this network, we acquire a little clue about the entity of the sample being studied. Random decision tree is a special type of decision tree where the questions, samples, or both are being selected at random. In a technical term, the features, samples or both are selected at random from a set of limited or infinite range of samples and features.

Training a decision tree is a propagative process, that is every node passes the learned knowledge to consecutive children nodes. This process is usually called g̈rowing the tree.̈ While nodes independently split, the growing of the tree is hierarchal. In other words, training a node at tree-level *tl* is essentially dependent on the completion of training parent node at tree-level *tl-1*. This top-down induction decision tree (TDIDT) algorithm relates back to the earliest tree algorithms[113]. The main advantage of this approach is efficiency[114]. Specifically, TDIDT generates computationally cheep classifier by pruning or skipping all siblings subtree in each level, reducing the search scope significantly (e.g. $2^{d-tl} + 1$ in case of binary). This is an advantage in cases where good performance with respect to low cost is necessary, though a potential disadvantage is that valuable knowledge in the skipped search space will not be explored. Figure 2.3 depicts the training process of random decision tree.

With random TDIDT algorithm in mind, random decision tree is built level by

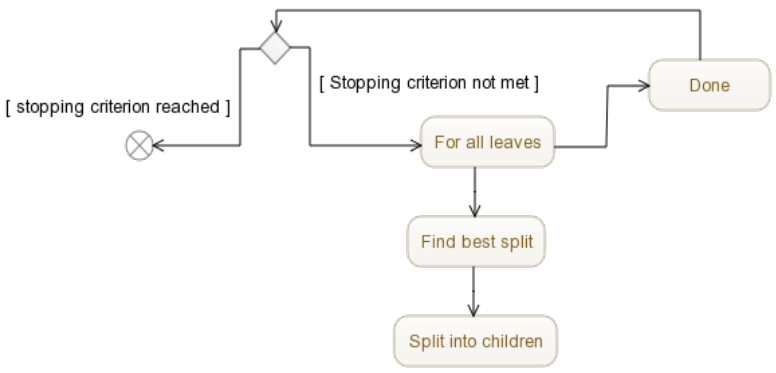


Figure 2.3: Narrate the story of training a random decision tree. Iteratively growing the tree by splitting leaves with best random splits until stopping criterion is met

level either in depth-first or breadth-first fashion[115]. Figure 2.2 shows an activity diagram that specifies the general algorithm of training a random decision tree. Generally, three issues must be considered[115]:

1. Split way: Multi or binary

2. Impurity measure: Gini index, entropy

3. Stop criteria

We gave a fair description above about the split way. The next subsections discuss impurity score and stop criteria in details

Impurity measure:

Internal nodes of a decision tree are split according to an evaluation of some splitting criteria, hence the best criteria to be selected is the one that can give the most informative split. Rokach divides the splitting criteria into univariate and

multivariate[116] and provides 14 different univariate measures. The most commonly used measure are Gini coefficient (also known Gini index) and information gain[116, 117, 118].

In the context of Random Forest, the aforementioned measures are used to decide on the best random feature $\theta*$ to split a certain sub-data D. In this study, we use information gain scoring function. Information gain, in the context of decision tree, is a quantitative measure of how much uncertainty is reduced by splitting a node according to particular attribute. The value is calculated by measuring the difference of the parents' Shannon entropy and the weighted sum of its children's entropies [119].

$$Gain(\mathfrak{D}, \mathfrak{D}') = H(\mathfrak{D}) - \sum_{v \in values(\mathfrak{D}')} \frac{|\mathfrak{D}_v|}{|\mathfrak{D}|} H(\mathfrak{D}_v) \tag{2.2}$$

In this study, we build binary trees. Each node is split into two nodes: left and right. The trees are trained with a set of discrete variables C (body parts or classes) and thus have a discrete probability distribution. Therefore, we can re-write Equation 2.2 as follows:

$$Gain(\mathfrak{D}, \mathfrak{D}') = H(\mathfrak{D}) - \sum_{s \in \{left, right\}} \frac{|\mathfrak{D}_s|}{|\mathfrak{D}|} H(\mathfrak{D}_s) \tag{2.3}$$

where

$$H(\mathfrak{D}) = -\sum_{c \in C} p(c \,|\, \mathfrak{D})\, log_2(p(c \,|\, \mathfrak{D})) \tag{2.4}$$

c is a target class and $H(\mathfrak{D})$ is the entropy in the class distribution of a certain node. Therefore, the gain by splitting $\mathfrak{D}$ into $\mathfrak{D}$' is the difference between the entropy in the class distribution of the parent node and the normalized weighted sum of the entropies in the left and right class distributions.

Stop criteria:

The depth of a decision tree d is the highest level number in the tree at which no further split exists or simply the longest path from the root to a leaf. The growing of the tree is applied recursively until stopping condition is met. Usually, there are global and local stop condition[120]. Global stop condition is meant to stop the growing of the whole tree, while the local is meant to stop the growing of a certain branch. The common stopping condition are[116, 121, 118]:

1. all instances in the training set belongs to a single value of y
2. the best splitting criteria doesn't meat a certain threshold
3. the maximum tree depth has been reached

The first two stopping conditions are, usually, associated with local stop; that is if they were not met at a certain leaf node, the node stops the growth of that particular path. The global stopping is usually associated with either all terminal nodes meat local stopping condition or the tree depth reach to a predefined level.

Stopping condition plays major role in the generalization of the decision tree. The decision tree is prone to over-fitting with loose stopping condition and prone

to under-fitting with tight one[116]. In order to solve this dilemma, pruning mechanisms have been proposed (originally by Breiman[122]). The aim of pruning is to remove noise and identify simpler trees with the lowest error rate. Pruning can take place while growing *(pre-pruning)* (e.g. stopping condition) or afterward by pruning some branches *(post-pruning)*. Esposito and Rokach provide a comparative analysis of methods for post-pruning decision trees in [123, 116] respectively. While pruning is proved to improve accuracy, the concept of Random Forest provides an alternative approach. With its underlying principle of tree averaging, Random Forest is less likely to over-fit, and would produce equally good results[124, 125]. However, Ren, in his recent paper [126], argues that a complete Random Forest without refining or pruning suffers from storage cost and still could be improved. He then proposes a refined model that resolves these shortcomings and show how it improves body part classification.

2.2.2 Random Forest

Random forest was first introduced by Ho [113] in 1995 to solve poor generalization issue by constructing multiple trees. Random forest, also called random decision trees or random decision forest, refers to the classification or regression technique that are an ensemble of multiple many or binary decision trees.

To classify or get a regression value for an input sample x, it must traverse each of the random decision trees τ of a random forest F.

The goal of Random Forest is to improve the accuracy of the classifier by ac-

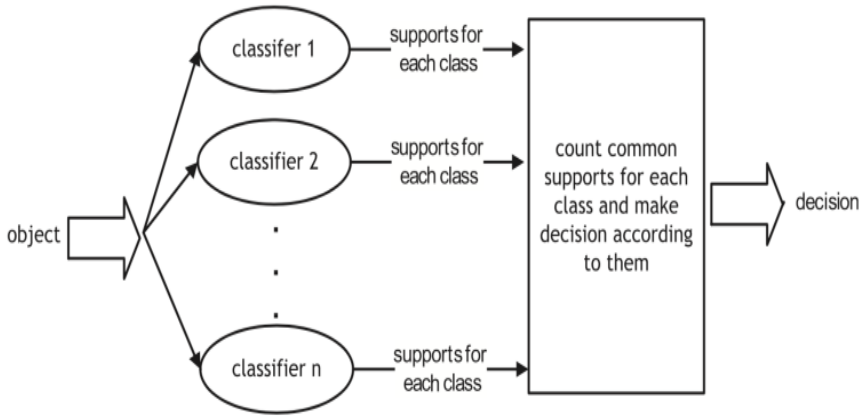


Figure 2.4: Architecture of the MCS which computes the decision on the basis of support function combination [127]

cumulating votes from several diverse base classifier. As far as classification is concerned, a majority voting is often used for final classification as illustrated in Kulkarni's Random Forest survey [118]. Kulkarni suggests that using weighted voting mechanism based on the strength that each tree demonstrates could however improve the overall accuracy. Giorgio provides a detailed analysis of how much theoretical reduction of error probability can be obtained using the weighted voting instead of the simple voting rule[128]. In [127], a textual analysis of the different combination (named fusion in the reference) techniques is presented. In this study, we are concerned with one of these techniques named support function fusion (SFF). The function evaluates the decision as a likelihood of a class derived from the class *posteriori* probability (Figure 2.4). The average *posteriori* probability for each class is thereafter averaged over all decision trees τ to give an overall confidence distribution. Equation 2.5 gives a mathematical representation of SFF. In chapter 5 we show how ensembling multiple decision trees in a random forest fashion improved the accuracy of the pose estimation compared to single

tree classifiers.

$$p(c \mid F, x) = \frac{1}{|\tau|} \sum_{i=1}^{|\tau|} p(c \mid l_i(x)) \tag{2.5}$$

where l_i is the leaf node reached in tree τ_i for pixel instance x

2.3 Pose Estimation Using Decision Trees and Depth Images

Machine learning algorithms have been widely used in the realm of image processing and computer vision. Inspired by antecedent literatures, Shotton compiles a body parts random forest classifier that is trained to utilize depth features and provide an estimation of the pose of human at every single frame. This algorithm gained a lot of attention due to its ability to produce an adequate estimation in real-time.

As discussed in Section 2.2, the idea behind randomized decision tree classifier is to build a tree-model that the item under study traverses through by moving downward left or right to reach a leaf node where its classification information is stored. As far as body parts classification is concerned, Shutton classifies each pixel in the testing images to one of a pre-defined set of body parts. The pixel traverses a tree based on its response against features stored in each node to reach a final landing leaf node. The following four subsections discuss the training phase including data samples, features, and objective function followed by a subsection

on prediction phase.

2.3.1 Data Samples

Training a classifier to be able to segment the human body into asymmetric function-based-parts requires capturing huge number of sample data. Moreover, the number of combinations of environment variations, clothing styles, and physical shapes of humans are endless. Yet, no training data that covers all these variation is available. Fortunately, however, image processing research has proposed several alternatives to overcome this issue[129]. Shotton uses real mocap data, retargeted to a variety of base character models, to synthesize large, varied dataset. The process is implemented in a rendering pipeline that iteratively samples a mocap frame positioned in a certain configuration and retarget it to a base model. Then, texture mapping and basic computer graphics techniques are used to render colorful and depth images respectively (Figure 2.5).

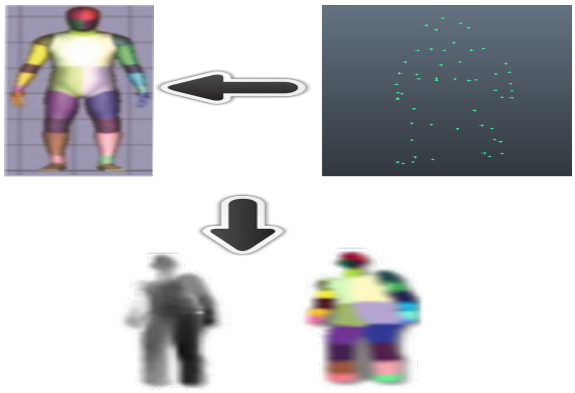


Figure 2.5: Synthesizing Training Data by retargeting mocap to base models and render a colorful and a normalized depth image from the resulting frames

Shotton uses total of 100K mocap frame covering variety of poses that human

body might be in in entertainment scenarios[6]. In addition, the frames cover other variations like camera noise, and point and direction of views. On the other side, a total of 15 base models are used to span various human shapes. This includes, sex, height, weight, age, clothing and other variations.

2.3.2 Body Labeling

The main contribution of Shotton algorithm is the introduction of an intermediate body parts representation. The main objective of parts definitions is to convert the problem from pose recognition into a machine learning object-class image segmentation problem. Moreover, body parts centered around joints positions are to lead to the prediction of the absolute location of these joints. Shotton divides the body into 31 segments with each body part colored differently (see Figure 2.5).

2.3.3 Deciding Features

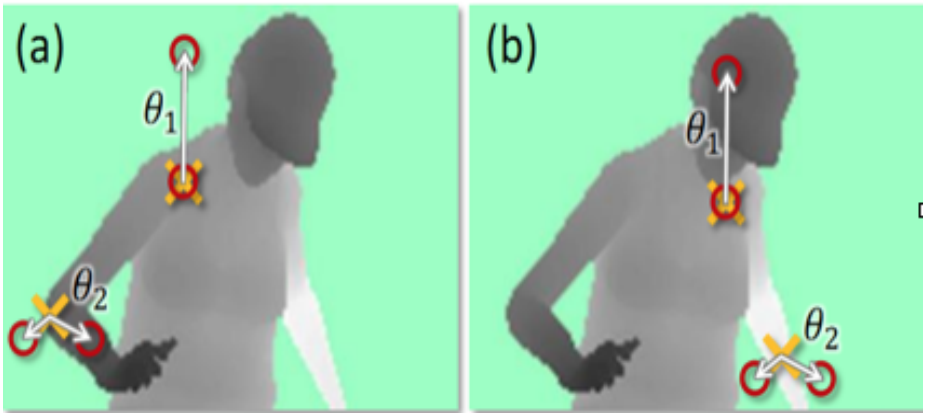


Figure 2.6: shows Shotton features. The red circles are two offsets u, v. The yellow cross is the sample pixel under study [6]

Shotton uses a simple depth pixel comparison features to classify each pixel

in the testing images to a certain body part (see figure 2.6). The pixel under study, denoted x, will branch left or right based on its response to Equation 2.6. These features although individually provides little knowledge about which part of the body x belongs to, in cumulative, they provide high certainty to a sufficient classification of that pixel.

As mentioned previously, the word random in random decision tree is attributed to the random selection of features and, sometimes, a random selection of training samples. Shotton random features denoted θ= (Φ,γ) are basically a combination of offsets and a split threshold denoted $\Phi = (u, v)$ and γ respectively. Figure 2.6 show 4 examples of Φ. Each two red circles depict a random offsets that are distanced u, and v from the pixel x depicted as yellow cross. θ_1 in (a) depicts a uni-offset that is either u or v are selected to be 0. Unlike the pixels (yellow crosses) in (b), both pixels in (a) will have a very large response to formula 3.6 considering that background pixels are given large depth values. If background pixels are assigned an identical large depth values, then the response of θ_2 in (b) will be 0.

$$f(x, \theta) = z(x + \frac{u}{z(x)}) - z(x + \frac{v}{z(x)}) \tag{2.6}$$

2.3.4 Training and Prediction

Shotton trains random trees using the same principles discussed in Section 2.2.1. Initially, a subset N_{ex} of random pixels (specifically 2000 pixels) are selected from each image in the synthesized dataset to form the training corpus $\mathfrak{D}$. Each pixel in

the corpus, denoted x, is a 2D pixel location in a particular image. The corpus acts as a fuel for constructing the random trees. Starting from the root, $\mathfrak{D}$ will be split and propagated downward until a leaf node is reached where $\mathfrak{D}_l$ will form then a confidence distribution of how likely a prediction-time x belongs to a particular body part c $\in$ C

The enumeration below describes how training a particular growing decision tree node $\mathfrak{i}$ is carried out:

1. Assume $\mathfrak{D}' \subset \mathfrak{D}$ is all training pixels that reached the node $\mathfrak{i}$
2. Sample a set of random features $\eta = \{\theta_1, \theta_2, \theta_3, ...\theta_n\}$.
3. Evaluate $I_{\mathfrak{D}'}(\eta) = I_{\mathfrak{D}'}(\theta) \; \forall \, \theta \; \in \; \eta$
4. Select $\theta^* = min(I(\eta))$
5. Split $\mathfrak{D}'$ with θ^*
6. Repeat 1 for $\mathfrak{i}_{left}$ and $\mathfrak{i}_{right}$

The objective in the training phase is to find the feature θ^* that best splits $\mathfrak{D}' \subset \mathfrak{D}$ that reached a particular node $\mathfrak{i}$. 2, and 3 in the enumeration above achieve that. In 3, the impurity score (see Section 2.2.1), denoted $I_{\mathfrak{D}'}$, is calculated for each randomly sampled feature. In our notations $I_{\mathfrak{D}'}$ is the second term in the right side of Equation 2.3. Subsequently, as in 3, the best feature is the one that can minimize the impurity score and thus increase the information gain. In step 5, $\mathfrak{D}'$ is split according to θ^* and then the process is repeated for the new children $\mathfrak{i}_{left}$

and $\mathbf{i}_{right}$. When a stopping criteria is reached, $\mathfrak{D}'_l \in \mathfrak{D}'$ for leaf node l is stored in the leaf node and the training of a tree $\tau \in$ Random Forest F is completed. Each tree in Shatton random forest is trained independently. Thanks to the fact that generating samples and features is fully synthetic and hence cheep.

At the prediction phase, a random forest $F = \{\tau_1, \tau_2, \tau_3 ... \tau_k\}$ estimates the pose of a human by classifying each pixel x in an input 2D image to a particular body part c $\in$ C. x starts at the root node and traverses the tree until it reaches a leaf node $l(x)$. The probability or the class distribution $p(c \,|\, l(x)$ stored in the node is used then to calculate a label for x. Basically, it is calculated as the relative frequency of training samples that reached $l(x)$ during the training. Thus, for each class, the probability distribution is calculated as follows:

$$p(c \,|\, l(x)) = \frac{\mathfrak{D}'_{l(x)}(c)}{\sum_{c' \in C} \mathfrak{D}'_{l(x)}(c')} \tag{2.7}$$

In a uni-tree forest, the calculated distribution is used to evaluate the label for x while in a multi-trees forest, each tree cast a vote and a combination function is used to determine the final distribution and hence the label. Shotton uses Equation 2.5 to calculate the forest final combined distribution $p(c \,|\, F, x)$ for x. The ultimate classification c^* is then calculated as the class with the highest probability:

$$c^* = max(p(c \,|\, F, x)) \tag{2.8}$$

2.4 Research Methodology and Solution Approach

In the previous sections, we provided a technical background of the concepts and tools which our study depends on. Kinect, synthetic depth and ground truth images, and random decision trees were used by Shotton in [6] to build his real-time human pose recognition framework. As we will see in Section 4.4, the framework doesn't accurately estimate the pose when exposed to non-western style cloth. Our target is to provide a remedy for such shortcomings. In this work, we focus on improving the pose estimation of human wearing Thobe using the same tools and concepts employed by Shotton. Specifically, we will be using synthetic dataset for training random decision trees, and eventually evaluate them with real dataset generated from Kinect.

The number of potential human poses is limitless, and thus it is almost impossible to compile a comprehensive dataset to train a system to generalize for unseen poses. In this study, we consider human wearing Thobe, a free floating dress, which exacerbates the problem. In order to overcome this issue, we consider estimating the pose from depth images only. The fact that depth images represent scene geometry information only allows us to synthesize depth images in virtual environment and use it for training. In virtual environment, we can synthesize large dataset with zero-cost. Shotton used 15 base-model character to synthesize more than a million pairs of depth and labeled dataset. In this study, we will use one base-model character of a human wearing Thobe. To label the data, Shotton

used direct texture mapping. However, in our case, and because the movement of the Thobe is chaotic, we will employ dynamic labeling, such that the label of each part of the model need to be calculated in each frame.

We will use the synthesized dataset to train random decision trees and random forests. As mentioned above, decision trees can be binary-split based or multi-split based. In this study, we use binary decision trees for two reason. First, binary split is easy to implement and work with. Second, we use Shannon entropy to identify the best split (see Section2.2.1) which works better with higher uncertainty in upper tree-levels. Consequently, splitting a node to more than two branches can cause early fitting of the data. Evaluating multiple split trees and other information gain measures is beyond the scope of this study.

Due to the randomness nature of the approach we are following, we will train large number of trees in structured and brute-force manner. The aim is to find a set of training parameters that generates trees with minimized prediction error and higher degree of generalization. To measure the effectiveness of the generated trees on synthetic test data, we will use quantitive comparative and self-evaluating measures. For real-dataset, we generate depth images using Kinect, and demonstrate our prediction analysis visually.

CHAPTER 3

POSE ESTIMATION OF HUMAN WEARING THOBE

3.1 Data Samples

As mentioned in the data samples section in the foundation, computer vision research are usually short when it comes to training data. Researchers usually propose and work with new data sets that suit their objectives. Moreover, the variations that we are studying in this thesis, have never been considered and we couldn't find a dataset that we can depend on. To overcome this issue, we developed a data synthesis process to generate our Thobe-based training dataset. The data is based on a human models similar to the one used by Shutton. However, our models introduce new complexities to the body part labeling process that make it improper to use straight texture mapping. The following three sections presents our method in solving this problem. The first section discusses the nature a our

base mode (Thobe-Based model). The second section illustrates how we used cloth simulation to animate the base-models. The third section introduces the body part labeling procedure that we used to label the base-model. The subsequent sections discuss, in order, the deciding feature that we have used, and our training and prediction procedure.

3.1.1 Thobe-Based Model

Thobe is a tailored fabric that is usually loose at the bottom and flexible to bend. We use a 3D base models of a man character wearing Thobe to represent real human-being (Figure 3.1). This idea is based on the assumption that the depth images will hide the unreality of the model and give a representative or abstract view of a real human-being (Figure 3.2). Since the focus of our research is to prove that Shotton prediction on Thobe can be improved by employing more representative models, we only consider a single base-model character. The character is created with a physique that represents a taut body person with a moderate tall. Our test cases match this configuration(Figure 3.2). In order to test for other configurations (e.g. fat, short, women), one needs to build another representative characters accordingly. As a matter of fact, Thobe position in reality looks different on a fat person than it looks on a slim one (Figure 3.3). Similarly, for a single pose, a cloth simulation would produce different depth images for each character that a classifier would have to account for. We used Autodesk Maya [130] for creating the base character.

Thobe Simulation

The look of the Thobe as a person is moving or changing poses can take endless forms. Moreover, the behavior of the Thobe is dynamic and highly sensitive to initial conditions that a small change in one condition in a particular state would yield a largely different behavior in later states. Depending on the occurring forces and the Thobe materialistic properties, a person could be standing still, while the lower end of the Thobe, unless stretched by the foot, is floating freely, creating random curvatures everywhere (Figure 3.4).

These uncertainties make it very difficult to come up with a single deterministic model to estimate the pose of the lower body. Alternatively, and similar to Shotton (see 3.3), we learn the pose of the person from synthetic data, rather than geometrically estimating it. Our data synthesis technique, however, differs from Shotton in that we had to handle the dress (e.g. Thobe) differently as it adds a new dimension to the pose that two identical character pose will have different dress position.

In order to cover for the wide range of possible dress positions and generate data that can be used to learn a generalized model, we employ basic animation techniques and cloth simulation. Cloth modeling and simulation has been widely studied in the domain of computer graphics and animation. In this domain, the appearance of the cloth throughout the simulation is more important than the physical accuracy[131, 132]. In other words, the precise physical reactions

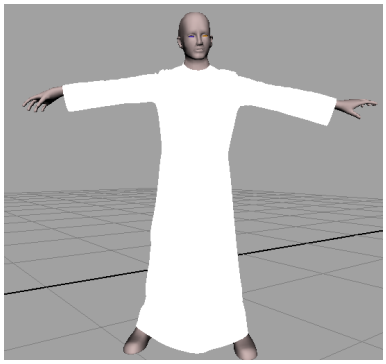


Figure 3.1: Thobe Model

Figure 3.2: Two depth images for real person (left) captured using Kinect, and base model (right) calculated using Maya renderer

Figure 3.3: Thobe looks different on fat people than it looks on thin people. Notice how the bottom end tends to flow with a higher degree of freedom at the front side in a fat person

Figure 3.4: Thobe made of heavy cotton material. Notice the curvatures on the right end hiding many vital details about the right leg pose

of the cloth object can be compromised as long as the simulation can achieve visual realism. In technical terms, we animate our base character in Maya (frame by frame), and then we employ Marvelous Designe[133] to create, simulate and animate the Thobe. For the purpose of this thesis, we created a simple 1200-frames scene where the base character performs simple moves.

While this technique helps to synthesize large dataset of images, it is worth to note, again, that the position of the Thobe depends heavily on initial conditions, and synthesized images generated from our 1200 frames are prone to failing in repressing unusual or turbulent initial conditions. However, to make up for the shortage of animation, and since we are using static depth images, and that our classifier would have to predict each frame independently, regardless of any environment or character variations, we tweaked the properties of the cloth's material to produce different scenes (see below). By doing this, we could reach to different Thobe positions for a single pose, and similar initial conditions (Figure 3.5).

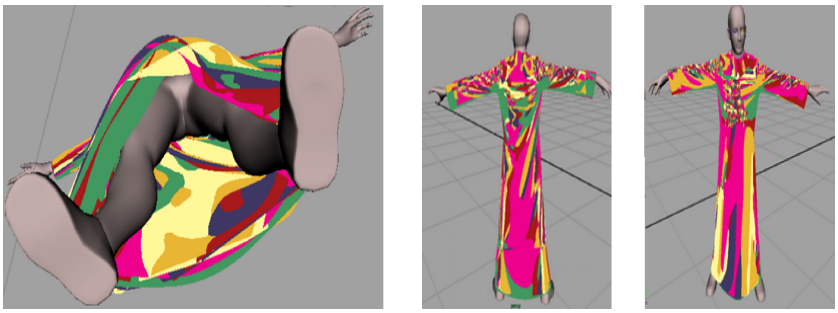


Figure 3.5: Figure 4.5 Different Thobe positions in the same timeframe and the same pose. There are six Thobes in the image, each is textured with a distinctive color. Each Thobe is set with different materialistic configuration. As appears from the orthogonal view in the bottom picture, each Thobe has different curvatures which together provide the trainer with different perspective for a single character pose. The two images at the top show a front and back view of the character. Note how they alternate in appearance.

3.1.2 Body Parts Labeling

Similar to women skirts, or bathroom robes, and unlike jeans, pants or sport shirts, Thobe has a free end at the bottom as we demonstrated in the previous section. Depending on the used material, the Thobe can have endless forms and shapes throughout the simulation (Figure 3.5). For all these uncertainties and complexities that the Thobe introduces to the pictorial look of the lower body, one cannot assume static mapping between model mesh segments and the adjacent Thobe faces in all frames. So, labeling training images would have had to be dynamic.

Similar to Shotton, we divide the body into parts Left Bottom Head, Right Bottom Head, Left Top Head, Right Top Head, Nick, Left Shoulder, Right Shoulder, Left Chest, Right Chest, Left Abs, Right Abs, Left Upper Arm, Right Upper Arm, Left Elbow, Right Elbow, Left Lower Arm, Right Lower Arm, Left Wrist, Right Wrist, Left Hand, Right Hand, Left Thigh, Right Thigh, Left Knee, Right

Knee, Left Leg, Right Leg, Left Ankle, Right Ankle, Left Foot, and Right Foot. However, we introduce a new body part: Thobe body part (TBP). This body part, figuratively and metaphorically introduced to the human body, is meant to represent the pixels-area in the lower body that doesn't belong to any body part or not within an adequate distance from a particular one to be annexed to it. The human targeted by this model can be thought of as a new creature that is a simi-human with an extra elastic and loose shell. The task here is to find the posture of this new simi-human creature. Practically, this model assumes that the Thobe is intrinsically a body part that needs to be identified at the prediction time. In the labeling process, each of the body parts mentioned above is given a distinct color. TBP is given a white color as it is the most common color of Thobe.

The nature of TBP necessitated a dynamic labeling. In other words, texture of the model has to be updated in each frame to correct the color mapping according to the new 3D location of the lower body parts. We developed a dynamic labeling algorithm that is executed at every frame to paint the lower Thobe in a way that roughly represents the spatial configuration of the hidden body parts. The algorithm tries to find pixels in the lower Thobe that can represent the hidden body parts beneath it. If a pixel cannot be linked to any body parts, it is said or linked to TBP. Specifically, it is labeled with TBP's assigned color (e.g. white in our case). Algorithm 3.1, shows a pseudo code of this procedure. As the Thobe is naturally tight to the body at the upper segment, we only apply this algorithm to

the lower part of the Thobe, namely the area below the torso. The upper part is labeled with a normal texture mapping. The coloring evaluation in Algorithm

Algorithm 3.1 Algorithm to label lower body

INPUT: LBF Lower body part faces
INPUT: THF Thobe faces
INPUT: ϵ maximum Euclidean distance threshold
REQUIRE: sort LBF by centroid.y
REQUIRE: sort THF by centroid.y

```
for all lbf ∈ LBF do
  c ⟵ lbf.centroid
  TFwT ⟵ [ thf ∈ THF: c.y - ζ ≤ thf.centroid.y ≤ c.y + ζ ]
  for all t ∈ TFwT do
    tc ⟵ t.centroid
    d ⟵ EuclideanDistance( tc , c )
    if d ≤ ϵ then
      color(t, lbf.color)
    end if
  end for
end for
```

3.1 is basically a two-level nested loops over a set of lower body part faces LBF (line 1) and a set of Thobe faces TFwT that lies within a Y-distance ζ from an lbf $\in$ LBF (line 3, 4). We use a simi-scan-line technique to traverse these faces from top-to-down. To do so, both of these sets of faces are sorted by the Y-coordination of their centroids (pre lines). Line 2 and 5 retrieve the 3D coordinations of the centroid value for a body part face $\in$ LBF and for for a Thobe face t $\in$ TFwT respectively. In line 6, 7, and 8, we calculate the Euclidean distance between the two centroids. If the distance is less than threshold, than that means the Thobe face is close enough to the body part and thus takes its color. Otherwise, the face will either be left uncolored, or be colored in future iteration by another body part. See Figure 3.6 below.

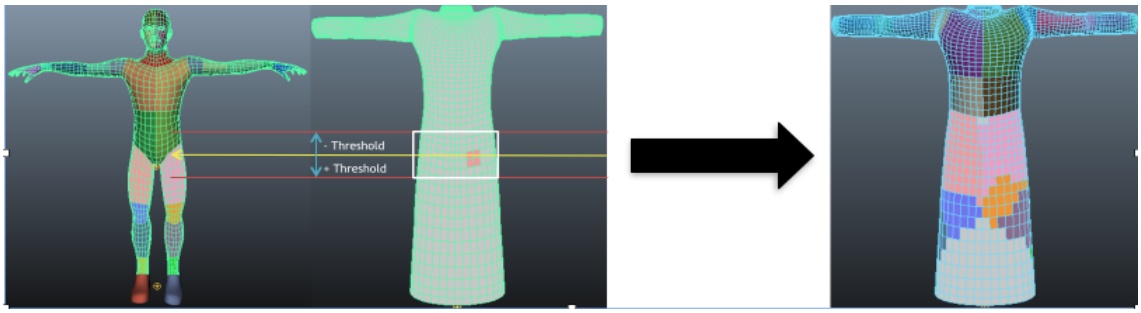


Figure 3.6: Illustrates the coloring technique. Image on the left shows the body mesh, and plain Thobe mesh. The yellow array is pointing to a body face on the thigh that is to be reflected on the Thobe. Red lines, and white box bounds a sub-set of the lower Thobe faces to be checked. Painted faces on the Thobe are faces that lie within ϵ Euclidean distance from the body face and thus took its color

The initial implementation didn't include the range step (line 3,4). In other words, we assume that the scan line technique guarantees that if a Thobe face is not within an ϵ distance from one higher body part in terms of Y-coordination, it cannot be within ϵ distance from any lower body part. However, this assumption can be easily challenged by a contradicting example. Specifically, a Thobe face can be top on Y-axis and within ϵ from a thigh face, but the thigh face is below a knee face. To overcome this issue, we introduced line 3, and 4. Also, to include all the possible candidates Thobe faces for a certain body part face, we set $\zeta = \epsilon$.

3.1.3 Rendering Synthetic Scene

There are several variations that can be utilized to build a sufficiently large dataset of synthesized images from a limited number of frames. These variations are inherently related to three main components in the rendering process: camera, model, and the dress. The variations that we selected have different purposes:

some of them are meant to feed the data corpus with different views for a single pose, while others are meant to increase the magnitude of the data.

Our major target in the rendering process and in selecting the variations was *inclusiveness*. Specifically, the dataset must be inclusive in two aspects: first, it must cover a wide range of main components' settings (e.g. model pose, and camera location and orientation) that we will be targeting at the time of testing, and second, it must consider the chaotic nature of the dress by having several images represent possible simulated positions of the dress for a particular pose. The fact that the definition of the feature proposed by Shotton, and being used here, includes a normalization by the depth of the sample makes the model 3D-translation invariant(see above). Thus, we set a constant distance between the model and the cameras. The following paragraphs give a summary for each of the variations we have used in our study.

Character Pose

Although we are concerned with static images, and unlike character wearing jeans or western-cloth, poses of our model must consider temporal information. In other words, if we set the character in a certain pose, then we have to set a proper Thobe position that suits this pose. However, this cannot be achieved without knowing how this pose came into place. For the purpose of this thesis, we create a story of 1200 frame where the character performs simple moves sequentially. The moves include: normal and stretched walking, slow running, jumping, hand movements, quadriceps stretch, and knee marching. To eliminate redundant poses,

we render only $\frac{1}{p_d}$ of the scene by discarding or skipping $p_d \in \{3, 4\}$ timeframes after each rendered frame.

Camera Position and Orientation

We set-up a studio-like environment for rendering. We place 16 cameras in the scene. The cameras are placed in two y-levels in a circular circumference such that they are within equal distances from each other. In the first level, we place the cameras at the y-level of the torso pointing straight forward to it, while in the second level, we place the cameras at a y-level above the head pointing downward to the torso. Therefor for every frame we produce 16 different images; each represents different point of view. A more thorough experiment would add a third level, where cameras are placed at the y-level of the foot.

Thobe Material

To simulate clothing, several dynamic parameters and physical properties of cloth (e.g. threads characteristics) must be incorporated into the model to produce the look and feel of a dress when it is worn by the character and interact with it. The simulation engine depends on these parameters to generate wrinkles and bulges and to calculate self-collision response and cloth-model collisions[132, 134]. Several material properties of different fabric patterns have been identified such as cotton, woolen, silk, leather, polyester and chemical fiber,etc [134]. During the simulation, these properties along with gravitational and collisions force determine the deformation of the dress.

In our experiments, we ran the scene in Marvelous Designer with 6 pre-set

different fabric patterns: chiffon, serge, pique, interlock, gabardine, denim, and cotton. Each of these patterns is associated with different buckling, stiffness, bending, and friction values. Figure 3.5 shows how these configurations produced different deformation in a simple T-pose. These dissimilarities is carried out in the simulation generating different Thobe positions for every single character pose and hence synthesizing multiple images for a single pose.

Rendering Process

We used Maya to render the scenes. While our images are basic and only include solid colors, nevertheless the rendering process was not straightforward. We use a basic surface shader material to give solid colors to the upper (static) and lower (dynamic) body parts. However, regardless what renderer we used, and what rendering configuration we set, the renderer always miss-colors some pixels or produces anti-aliased images that introduces extra colors to the image. There are two main ways to handle this issue: handle it during the training and prediction by ignoring pixels not in our color range, or add a post-rendering technique to fix these pixels. We used the second method, which should be cleaner, and would generate reusable, and sanitized images that save some time during training. Thus, our rendering process is divided into three stages: pre-rendering, rendering, and post-rendering. An image to be rendered has to pass through all of these three stages.

Pre-rendering

In the pre-rendering stage, we mainly update the scene to a new timeframe (start-

ing from 1), and then apply Algorithm 3.1 to label the character accordingly. As described in Algorithm 3.1, the labeling algorithm requires two inputs: list of body part faces and list of Thobe faces. Technically, these two lists contain the 3D locations of the 4-vertices that make each face. After a new timeframe is set, these lists must be updated with a new 3D location for each vertex. Another possible approach, that we didn't implement, is to parse the entire scene from the Maya cash exported from Marvelous design, and load it into memory. This way, no needs to call Maya commands in every frame (see code details below)

Rendering

We tested several Maya rendering engines. Our intention was to build a dataset of images with a range of 32 colors only, each of these colors represent a body part (see Section 3.1.2). Unfortunately, all engines we have tested failed to fulfill such requirement. As mentioned above, images are either heavily anti-aliased or have wrong colors in random places. Eventually, we selected Maya Hardware renderer as we found it to be the one with the least coloring errors. It is worth to note that we used the legacy version of Maya Hardware renderer not Maya Hardware renderer 2.0. Figure 3.7 shows a snapshot of the rendering configuration that we used. The images are stored in ".iff" file format. An .iff file is an image interchange format used by Maya. These images can have 8 or 16 bit per channel RGB, and optionally Alpha, and can optionally contain a 32-bit floating point depth map. Practically ".if" format combines ground truth images, depth data, and inferred depth visualization images in one file. Although we extract these images in the

post-rendering stage, we exploit files in this format when transferring the data, and as backup files.

After rendering is completed, post-rendering script parses the ".iff" file

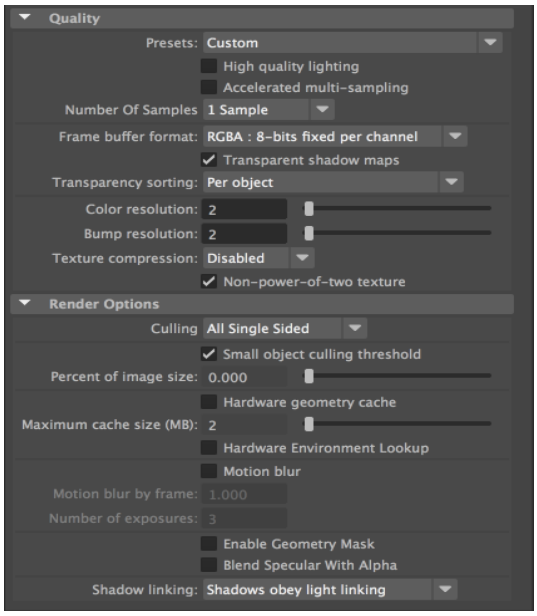


Figure 3.7: A snapshot of the rendering settings that we used.

and fissions it to three ".png" images: x_ground_trueth.png, x_depth.png, and x_depth_visualization.png. The depth image is not viewable; its dimensions are identical to the ground truth image but with a single value in each pixel represents the distance between the camera and the most closest 3D word hit. It can be thought of as the z-buffer. The depth is presented in mm (millimeter) and is in the range between 1000-8000. The depth visualization image is generated to give a visual representation for the depth image. It is a gray-scale image that assigns pixels a color density [0-255] based on its relative depth value. The pixel with the closest depth is assigned black color (255). Finally, all images are rendered with dimensions: 960X540

Post-rendering

To provide the training code with sanitized dataset of images that require no further checking, we fix and adjust our images. Post-rendering stage mainly involves two tasks: cropping out background, and fix miss-colored pixels. We achieve both tasks while extracting the images from the ".iff" files.

The first step in the post-rendering stage is to crop the images to the character area. The dimensions of the rendered images are greater than the longest and the widest range that the character can reach. We simply eliminate the empty space of the images' by identifying first non-background pixel in each side, and chop a cutout of the character. The extended dimensions and the cropping is just a precocious exercise to ensure that the character is the only visible object in the scene.

The second step in the post-rendering stage is to fix miss-colored pixels. The renderer we used, while configured with the lowest quality settings, still generates anti-aliased images. Anti-aliasing is a term used to describe an image filter or a software process of making the edges of graphic objects or fonts smoother. While this filter is so powerful and has an advantageous effect on the spatial resolution of images, it is a complete drawback in our case. In order to fix the anti-aliased images, we apply basic recursive algorithm to find and color miss-colored pixels with the closest correct color (CCC). Algorithm 3.2, shows a pseudo code of this algorithm.

Rendering Process Altogether

Algorithm 3.2 Closest Correct Color

INPUT: im image array
INPUT: (x,y) x,y location of the miss-colored pixel
INPUT: (w,h) width & height of the image
INPUT: offset neighbor offset

```
for all (i,j) ∈ [(x ± offset,y ± offset)] do
  c ⟵ im[i,j]
  if c ∈ [correct colors] then
    NCC.append(im[i,j])
  end if
end for
if len(NCC) == 0 then
  recurs(im,(x,y),(w,h),offset+1)
end if
return most_frequent_element(NCC)
```

The main advantage of using Maya renderers is the ability to interact with the scene programmatically using Maya's python API and interpreter (mayapy). The API provides a standalone module where we could open the scene file and interact with its content. The entire rendering pipeline including pre and post rendering is implemented in python.Algorithm 3.3 depicts the full rendering processes. Lines 2-10 inside the for loop are the entry points for all the stages of the data sampling process described above. "label_lower_body" in line 8 is a function call for the implementation of Algorithm 3.1 whereas post_render is a function call for post-rendering stage. "update_faces" in line 4,5 is a function call to update a list of vertices' 3D locations. As the scene progresses in the timeframe, the 3D location of the objects in the scene are changing, and thus code lists that track the vertices of the objects need to be updated. In each iteration, we reset the lower part of Thobe color to white in "whiten()" in line 2.

Algorithm 3.3 Full Rendering Processes

INPUT: start starting timeframe number
INPUT: end ending timeframe number
INPUT: jump number of timeframe to jump per iteration
INPUT: bodyfaces base-model mesh faces
INPUT: thobefaces Thobe mesh faces

```
for i ∈ [start:end:jump] do
  whiten()
  set_current_timeframe(i)
  update_faces(bodyfaces)
  update_faces(thobefaces)
  sort_by_y(bodyfaces)
  sort_by_y(thobefaces)
  label_lower_body(thobefaces,bodyfaces)
  MAYA.HardwareRender()
  post_render()
end for
```

3.2 Deciding Features

In machine learning, finding a feature set that can best represent a particular entity plays a major role in improving the perception of the system. In the realm of computer vision and robotics, several visual feature have been employed for detecting, recognizing and grasping objects. The selection of the visual feature is highly associated with the visual data description (e.g. Silhouettes, RGB-D, Depth...etc). Considering object segmentation and classification where RGB-D data are used, many detection algorithms employs image channels difference (e.g. R, G, B, A, and D) inspired by Lepetit[135] and [136].

In Section 2.3.3, we discussed how Shotton uses depth comparison features. Stuckler implements both RGB channels and depth channel differences [137]. Also, he used average region difference instead of a single pixel difference. In this the-

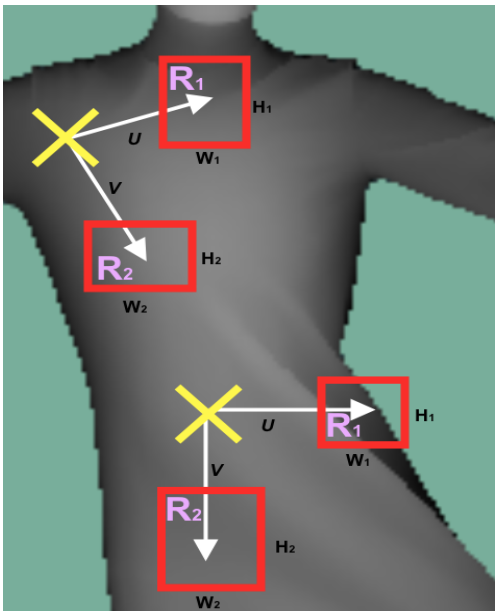


Figure 3.8: Average region depth features. A feature response to the pixel at the yellow cross will be calculated as the difference between the average depth of R_1 and R_2

sis, we focus on estimating the human pose from depth images only. However, we resemble both Shotton and Stuckler feature by evaluating both single pixel depth difference and average pixel-region depth difference. Figure 3.8 shows two example of average depth feature. In each example, the two offsets u,v, depicted as white arrows, the two widths (w_1, w_2) and the two heights (h_1, h_2) are selected at random. R_1 and R_2 are two regions centered around u,v with dimensions $(w_1 X h_1)$ and $(w_2 X h_2)$ respectively. The response to the query pixel depicted as a yellow cross is calculated as the difference between the average depth of region1 and region2 as in Equation 3.1. Because both regions are on the body, the top pixel in the figure will give a small response since. The bottom pixel on the other

hand will give a very large response since there are few background pixels in R_1

$$f(x,\theta) = \frac{\sum_{p \in R_1} z(p)}{|R_1|} - \frac{\sum_{p \in R_2} z(p)}{|R_2|} \tag{3.1}$$

To achieve depth invariance (e.g. distance from camera) when selecting the parameters of the features, we use the normalization in Equation 3.2

$$R(x) = \left(x + \frac{(u,v)}{z(x)}, \frac{w_{1,2}}{z(x)}, \frac{h_{1,2}}{z(x)} \right) \tag{3.2}$$

CHAPTER 4

EXPERIMENTS AND DISCUSSION

This master thesis focuses on evaluating and improving the current state-of-art algorithm for human pose estimation from depth images with unforeseen clothing variation. Specifically, we try to improve the body part labeling introduced by [6] of a human character wearing Thobe. We achieve this by employing decision trees trained with synthetic data. The main contribution of our work is to introduce a new base model and labeling technique to synthesize Thobe-Specific dataset that can be used for training the decision trees. In Section 4.1, we describe the public and the introduced datasets that were used in our evaluation. Section 4.2 discusses our experiments setup and environment (e.g. training, machines spec...etc). Section 4.3 gives an overview of the assessment metrics that are used to measure the performance of the classifiers. Section 4.4 evaluates the performance of Kinect pose estimation framework along with its underlying algorithm described in[6] on

real human wearing Thobe. In Section 4.5 and Section 4.6, we demonstrate our experimental results with the introduced model and labeling algorithm. We show the results of manipulating the different training parameters (e.g. tree depth, sampling type, region size...etc). In Section 4.7, we discuss the threats to the validity of the experiments and results.

4.1 Dataset

The availability of datasets that can be used to evaluate and compare methods for a certain problem contributes to the evolvement and flourishment of its solutions. This can be quantitatively proved by looking at the number of citations for papers that introduce new comprehensive or problem-specific dataset. For example, in the context of imaging and computer vision, see[138, 139, 140, 141]. However, as for human pose estimation, many solutions exist in the literature, nevertheless only a few deal with the problem using solely depth images and provide adequate testing data. Moreover, the lack of dataset in our case is exacerbated by the fact that none of the available pose estimation datasets target this type of variation and environment, let alone providing ground truth measurements. The following two sections discusses in order the public dataset (that we are aware of), and our introduced synthesized Thobe dataset.

Figure 4.1: Denil's depth, ground truth body parts and predicted body parts [147]

4.1.1 Public Dataset

Pose estimation is often given with respect to a skeleton. In other words, a pose is represented by a set of points that are connected together to form a skeleton of the shape under study. Similarly, human pose estimation is represented by a human skeleton that is constructed from 12-32 points depending on the data resolution. There are several number of public datasets that provide RGB-D images for different human actions and activities recorded for different recognition purposes[142, 143, 144, 145, 146]. As far as we know, all the public datasets provide skeleton-based ground truth data.

This thesis focuses on recognition of body part from depth images only. Therefore, the ground truth representation for our case should be an image with solid-colors depicting body parts. Unfortunately, neither the training dataset used by Shotton or the base-models are published. Luckily, Denil publishes a set of 2000 frames that are labeled with 20 colors (19 body parts + 1 background) [147]. We used this data, first, to reproduce the work of Shotton and, later, for comparison purposes. Figure 4.1 shows an example from this dataset.

4.1.2 Introduced Thobe Dataset

Synthesized

To kick off a series of future projects aiming to provide optimum estimation of human pose with difficult dress variations, and to compensate for the lacking of training and testing data, we used data synthesis approach discussed in chapter 4. Initially, we followed the body segmentation provided in [6] which divides the body into 31 segments. Later, however, we simplified the segmentation by merging the lower body parts into two and four segments. The following paragraphs discuss the 3 segmentations in details.

2-colors

In this dataset, we segment the body into 26 parts. The upper body part is segmented according to definitions in Section 3.1.2. The lower body part (excluding the feet) however, is divided into two parts only: Left Lower Part, and Right Lower Part. Each of these two segments encompasses the thigh, the knee the leg, and the ankle. The labeling method (Algorithm 3.1) presented in chapter 4 will not change, but the input colors for the character lower body parts (Figure 3.6 will be combined into two colors only (left and right). Figure 4.2(a) shows a set of examples of this dataset. The main purpose of this dataset is to test the ability of the system to differentiate between left and right with in presence of Thobe.

4-colors

In this dataset, we segment the body into 28 parts. Similar to the previous dataset, we divide the lower body parts in 4 segments: 2 Left and 2 Right. This

segmentation is adopted in [147, 126] as well. This segmentation, in fact, is more appropriate for two reasons. First, the lower body part consists of two big bones on each side: Femur (Thigh Bone), Tibia and Fibula(the leg bones). These bones construct two complete distinctive body parts: thighs and legs and hence can be represented by 4 colors. Second, Thobe hides all the distinctive features of the knee and the ankle, and in many cases (e.g. T-Pose) it is very difficult to estimate their specific locations without considering the geometrical nature and properties of a human being. Figure 4.21(b) shows a set of examples of this dataset.

8-colors

The 8-colors segmentation divides the body into 32 parts. It was first introduced by Shotton in [6]. The upper and the lower body parts are segmented according to definitions in Section 3.1.2. Shotton used this segmentation as intermediate step towards identifying the 3D locations of the body joints. Finding the joints is beyond the scope of this thesis. However, in such goal, having a distinctive color for the knees and the ankle plays a major role in estimating the correct 3D location of their associate joints. Figure 4.21(c) shows a set of examples of this dataset.

Real

In order to test our approach on real case scenarios, we recorded a set of RGB-D images using Kinect v2. Three characters were involved: short, medium and tall. Each of the characters performed a set of pre-defined moves: T-Pose, dancing hand, walking, jumping, stretching, quadriceps stretch, and knee marching. The

characters were recorded performing these moves twice: first wearing jeans, and then wearing Thobe. A set of 100 frames/character/dress were captured and then filter and reduced to a total of 100 frames. Figure 4.22 shows some examples of this set.

4.2 Experiments Environment

4.2.1 Tools

In this section we describe the tool that we build to synthesize the dataset, and the canned tools that we used for training and prediction.

Synthesizing Dataset

In order to generate the datasets discussed above, we used Maya[130], Standalone Maya Python Interpreter (mayapy), and Marvelous Designer (MD) [133]. First, we created a a base-model using Maya, and then animate it frame by frame. In parallel, we used MD to design the Thobe. We export the motion from Maya to MD to generate Thobe simulation cache. We import the Thobe and the simulation cache back in Maya to have a fully animated Thobe-Model. For rendering and labeling, we implement Algorithm 3.1 on a standalone Python program that can be run with (mayapy). The raw material for synthesizing the dataset can be downloaded from [148]

Kinect evaluation and Real Depth Images Dataset

To capture the real dataset presented above, and to evaluate the performance

of Kinect joint detection framework (see Section 4.4.1), we used Kinect Studio [149]. Kinect Studio is a utility application that you can use to preview Kinect sensor array data, record and play eXtended Event File (XEF) files, control the timeline position, and select 2D or 3D views. Kinect Studio APIs enable you to develop custom tools, to record and play back body data using XEF files.

Training and Prediction

Since we were using shared resources, fast and parallel implementation were required to train our random trees with different combination of parameters' values in relatively short time. Moreover, the parameters values are definitely directly proportional to the training time[150]. In other words, increasing the number of thresholds or features sampled per node should increase the number of computation needed and thus increase the runtime. Another factor that increases the runtime substantially is the depth of the tree. Each level in the tree doubles the number of nodes which thus mandate sampling new set of features and thresholds for each node. However, it is shown that in the context of image classification and segmentation increasing the depth of a decision tree has a positive influence on its prediction accuracy as long as there are enough samples [6, 151]. In a big-O notation, the theoretical time complexity for training a random forest of size T and depth d is $O(T \cdot d)$ [115]. The theoretical time complexity for training a single node with $|\Phi|$ number of random features, $|\gamma|$ random thresholds, and $|D|$ samples is $O(|\Phi| \cdot |\gamma| \cdot |D|)$

It is often that for any speed improvement gained, a less memory space effi-

ciency is resulted. Looking at our environment specification that is powered with large RAM capacity, we decided to compromise space complexity in favor of speed. Instead of implementing the training and predicting system from scratch, we capitalized on the implementation of Waldvogel in [152]. Waldvogel implements an accelerated random forest on CPU and GPU for object-class image segmentation with visual features that operates on color and depth information of RGB-D images.

In our experiments, we eliminated the code part that is related to color features and only used depth features. We used the CPU acceleration mode for training and GPU acceleration mode for prediction. As far as training is concerned, we selected CPU mode for two reasons: it runs faster in our environment and we have fewer nodes with powerful GPU. Therefore, running on CPU allows us to train each tree of a random forest with a certain set of parameters independently on different nodes. On the other hand, the prediction is much faster and more appropriate to run on GPU. By loading the trained trees to the static memory of the GPU, a stream of images can be classified in pixel-per-core fashion that can achieve less than 45 ms per image[152]. While comparison of runtime is beyond the scope of our thesis, we show in Figure 4.23 how the execution time increases as we increase the number of features, threshold, and dataset size.

Processor	Intel Xeon CPU E5-2680 0 @ 2.75
Number of cores	16
RAM	250GB
Cache Size	20480
GPU	nVidia Quadro 6000
CUDA Version	6.5

Table 4.1: The specifications of training and testing environment

4.2.2 Environment Specifications

We ran our experiments on a cluster of 64 high-end machines that have identical spec shown on Table 4.1

4.3 Assessment Metrics

In the following sections, we will be using standard machine learning metrics to evaluate the performance of the classifiers. In this section we give a description of each metric and what will it be used for.

Class Accuracy

Class accuracy is the percentage of correctly labeled pixels for a given class, and consequently the total average class accuracy is the sum of averages over the number of classes. This metric is used to show how much of each class were detected regardless of the false-positive rate. We use this metric to compare the performance of the trees as we manipulate the training parameters.

Pixel Accuracy

Pixel accuracy is defined by the number of all correctly classified pixels over

the total number of pixels in the image. Pixel accuracy gives equal weight to every pixel and thus is expected to be more conservative with the average value. The pixel accuracy is also used to compare the performance of the trees as we manipulate the training parameters. This measure can help to ensure that the average class accuracy is coming from distributed class accuracy (i.e. the standard deviation is low).

Precision, Recall, and F-score

The precision is a measure to show how many of selected pixels are relevant to the class or body part. The recall measures how many relevant pixels are selected. The recall value is equivalent to the class accuracy. The f-score is measure to average the precision and the recall value to provide a single value for the overall performance[153]. When we use these measures, we consider each body part to have an independent classifier and calculate the numbers accordingly (i.e. we calculate the precision, recall, and f-score values for each body part). Unlike the previous two metrics which are used for comparison between classifiers, the f-score analysis is used to quantify the effectiveness of certain tree or classifier.

Confusion Matrix

The confusion matrix is a table that is used to show the performance in supervised machine learning. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice versa) [153]. We use confusion matrix to visualize the confusion that arises between body parts. This matrix allows to see what each body part is being

predicted as or being confused with.

ROC Curve

The roc curve is another way to show the overall performance of a classifier. When analyzing the performance presented by a ROC curve, the points on the left sides are considered to be conservatives, they make decision only with strong evidence, and hence they make few false positives (in our case, they do not say this pixel belongs to X body part while it is not). On the other hand, staying near the x-axis, means the classifiers are making less true positives as well[153]. We use the ROC curve to measure the performance of classifying each body part. So, each point in the curve represents the performance of a body part classifier.

4.4 Pose Estimation Using Western-Style Classifier

4.4.1 Evaluation of Kinect v2

Before we started our project, we conducted a live test to Kinect v2 body tracking framework[154]. This test confirmed our hypothesis that the current underlying classifier used in Kinect is not applicable with characters wearing dresses that cover the lower body, and hence is not applicable with character wearing Thobe. To begin with, we used [155] and exposed Kinect to a character wearing Jeans and T-Shirt. As expected, the performance was objectively good. As can be noted in figure 5.5(a), Kinect body tracking was able to connect a smooth skeleton between

the estimated upper body joints in all cases. However, it fails horribly to estimate the lower body joints positions with poses that include stretching or self-inclusion (see 2, 4, 10 left to right, up to down). Second, we exposed Kinect to a person wearing Saudi white Thobe. As can be seen in Figure 4.24(b), the results were abnormal. The simple experiment, with exact same poses, gave incorrect detection of the lower body in all scenarios. These experiments were carried out numerous times, and the results in each were randomly incorrect. Figure 4.25 shows the skeleton predicted for two T-Poses, one with the lower Thobe on a high peak and another while its flat (left/right respectively). It can be noted that the system ignored the clothing fact, and interpreted the peek as a knee (left), and the flat part as outstretched legs and thighs.

4.4.2 Body Labeling Using Western-Cloth-Based Classifier

This study is based on the work of Shotton, and thus it was important to have a full understanding of his method. As mentioned previously, it is unfortunate that non of Shatton data related to this research is published. However, we sought to replicate the experiments by employing both Denil's dataset [147] and a dataset generated using our own undressed-based-model.

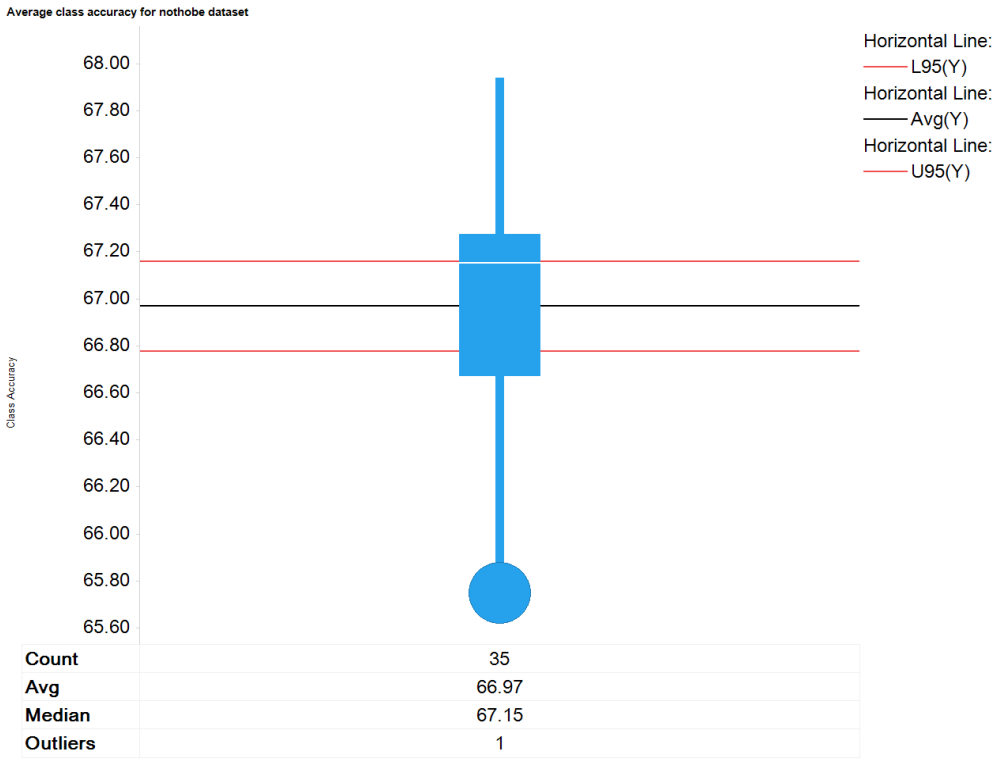
The difference between Daniel's dataset and ours is the number of body parts and dataset size. Danil's divide the body into 20 segments, while we follow Shotton's segmentation which divides the body into 31 parts (Figure 4.2). Moreover,

Figure 4.2: Shows an example of a synthesized image. On left: an example from Danils' dataset. On right: an example from our dataset

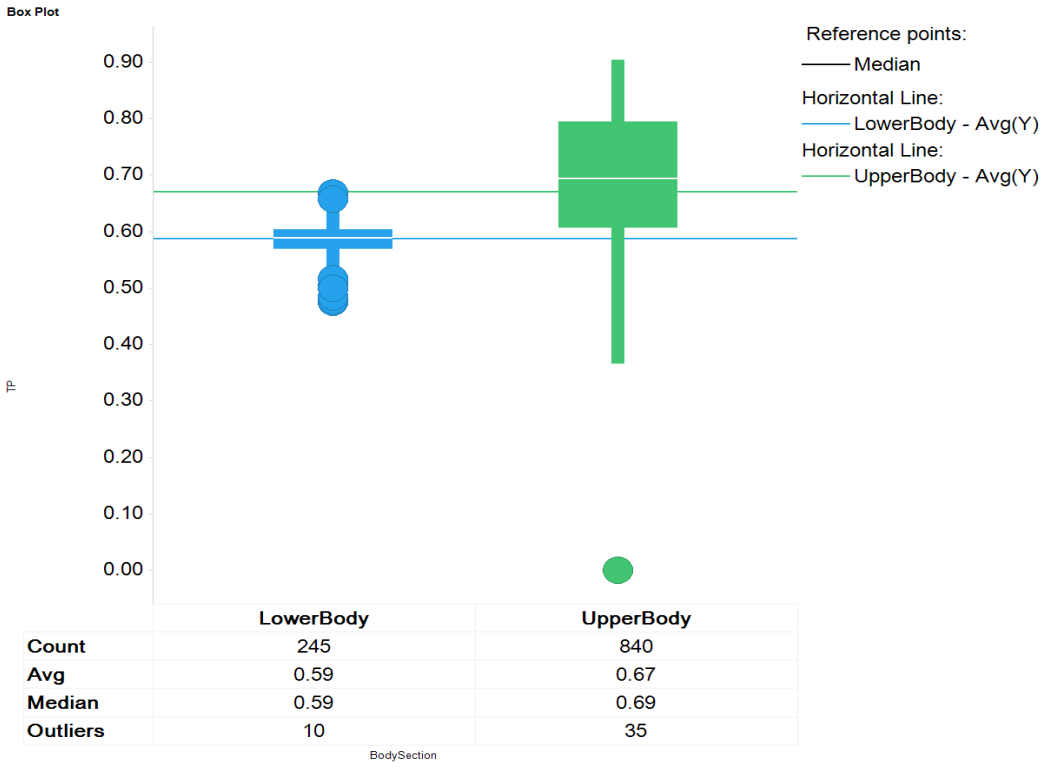
Daniel uses 2000 images for training, while we use more than 10000 images.

Using our dataset, we trained a set of 35 decision trees, each with the following parameters: 20 levels depth, 2000 training example pixels per image, 1800 candidate features, and 50 candidate thresholds per feature. The average class accuracy for the trees is 66.97% (Figure 4.3(a)). In other words, on average the trees were able to estimate 66.97% of each class correctly. In Figure 4.3(b), we show a distinct average for the lower and upper body parts. While we expect the lower body to have a lower accuracy for Thobe-Based classifiers, Figure 4.3(b) shows that Western-Cloth-Based classifiers suffers the same shortages also. The lower body parts scored 59% accuracy on average compared to 67% for the upper body parts.

As mentioned above, random forest is a powerful tool to improve the performance of decision-based classifiers. Figure 4.4(a), shows the average class accuracy neighbored by the upper and the lower 95% confidence interval with an increasing



(a)

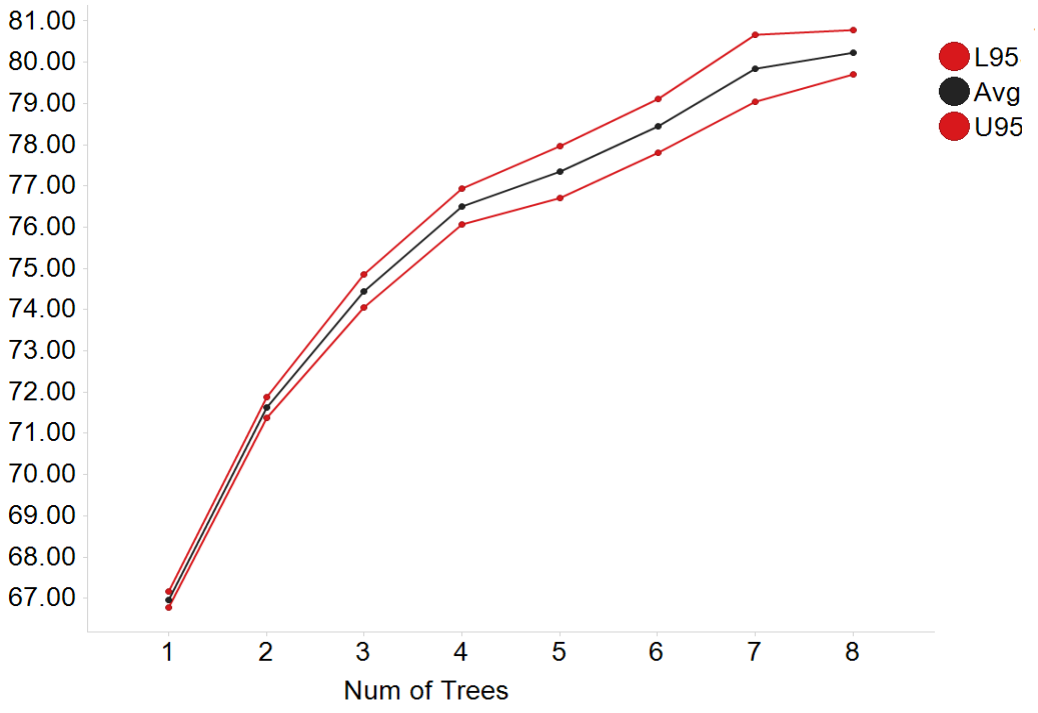


(b)

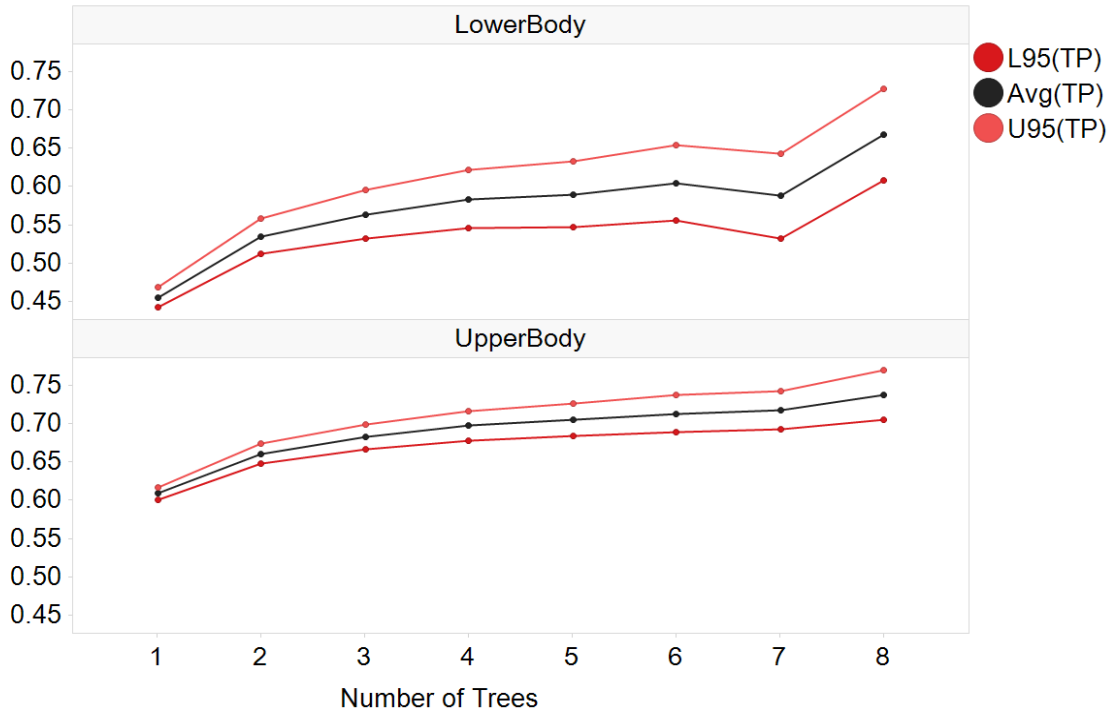
Figure 4.3: (a) Average class accuracy for 35 decision trees trained and tested with Western-Cloth-Based dataset. (b) A distinctive average class accuracy of the classifiers for the upper and the lower section of the body.

number of trees. The graph shows that the average class accuracy of the WCB classifiers improved dramatically as the number of trees increases. The divergence of the confidence interval from the average on larger values of trees is due to less number of trials (e.g. 40trees/8= 5 trials). With 8 trees, an ensemble classifier could achieve more than 80% class accuracy. Moreover, Figure 4.4 (b) indicates that classifiers achieve greater accuracy for both the lower and the upper section of the body as the number of ensemble trees increases. Embedded in Figure 4.4 (a), we show a set of triplets of synthetic depth images, ground truth images, and predicted images of an 8-trees random forest

For real test, we investigated the effectiveness of our classifiers using real dataset discussed in Section 4.1.2. Figure 4.5 shows an example of synthetic outcome (a) and real outcome (b) of this classifier. As can be noted from (b), while not very clean, the classification of real test is quite satisfactory. In other words, with our base-model that looks quite different from the real test, the classifier was able to generalize and provide discernible segmentation of the body with roughly 50% true positives. Having reached this accuracy, we were at confidence level to curry on and add Thobe to the base-model.



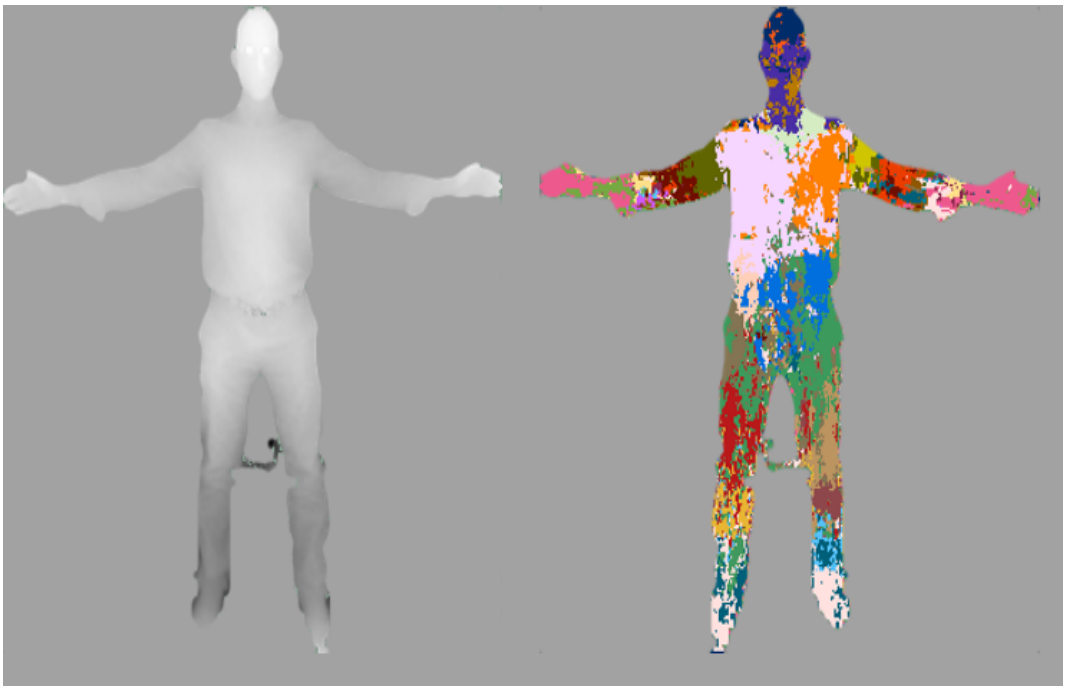
(a)



(b)

Figure 4.4: (a) average class accuracy versus number of trees in random forest classifiers. (b) average class accuracy per body section. Embedded in (a): triplets of synthetic depth image, prediction, and ground truth images.

(a)



(b)

Figure 4.5: (a) Triplets of synthetic depth, prediction, and ground truth of the T-Pose. The prediction of this image achieve 90% class accuracy. (b) A real depth image of a T-Pose captured by Kinect and the associated body part classification proposed by the same classifier in (a).

4.5 Pose Estimation Using Thobe-Based Classifier

This section presents our experiments and results on estimating the pose of human wearing Thobe using single random decision trees and ensemble random forest classifier trained with Thobe-Based training data. We evaluate our classifiers with the synthesized and real testing data presented in Section 4.1.2. For the synthesized data, we use average pixel accuracy and average class accuracy measures for benchmarking. For real data, the accuracy is assessed by visually comparing the predicted images.

In our experiments, we investigated the effect of training parameters on the accuracy of the classifier. The subsequent sections explain and summarize the results acquired by manipulating each of these parameters. All of the results shown below are related to the 4-colors dataset. The final section, however, compares the best classifier generated from each of the three types of datasets. We leveraged our resources capacity and trained more than 600 trees searching searching for a set of training parameters that takes less training time, yet achieve average accuracy. The resulting parameters are shown in the table below. These values are used afterwards to generate the graphs shown in the discussion below. In each section, we manipulate the parameter under study, and set the values of the other parameters according the values in the table. For each parameter, we train 10 trees and take the average (e.g. 10 for 800 features, 10 for 1000 features...etc). Because our sample size is small, in all the following graphs, we

Parameter	Value
Depth of Tree	20
Number of Features	800
Number of Thresholds	50
Maximum Offset Size	300
Number of trees	3
Maximum Average Region Size	3
Sampling Type	Class uniform
Training Size	23K

Table 4.2: Training parameters values. In each experiment, we manipulate one of these parameters and keep the other according to the values in the table

show the lower and the upper 95th confidence interval along with each prediction

value

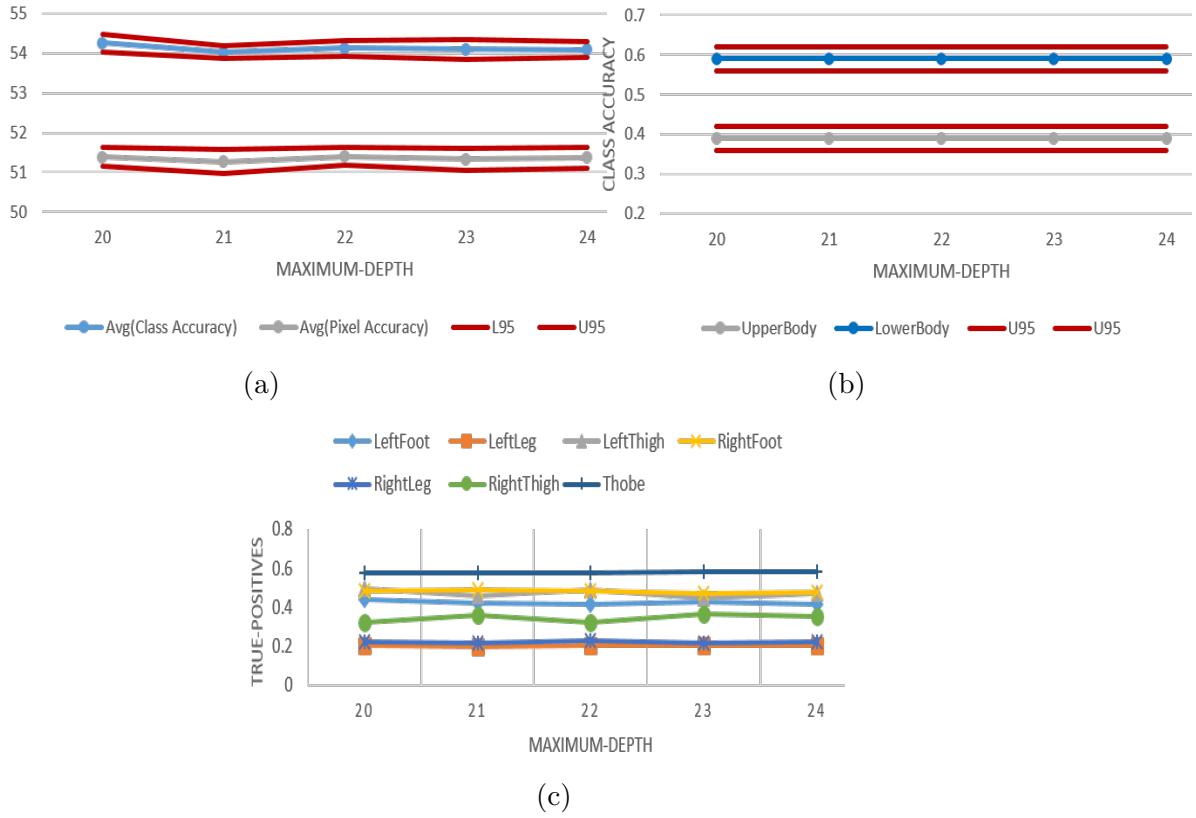


Figure 4.6: (a) average class accuracy versus increasing depth. (b) average class accuracy per body section (c) Average true positive for each body part in the lower section including TBP

4.5.1 Depth of Tree

The depth of the tree is the maximum level that a random decision tree can grow to. The line graph in Figure 4.6(a) shows the impact on the average class and pixel accuracies of 10 random trees in response to increasing the maximum tree depth. It demonstrates no gain in the overall average by growing the trees beyond 20 levels. The chart at (b) indicates the same fact for both the lower body, and the upper body. This implies that at depth 20, a high confidence is reached and the decision inferred doesn't change by further splitting the distribution.

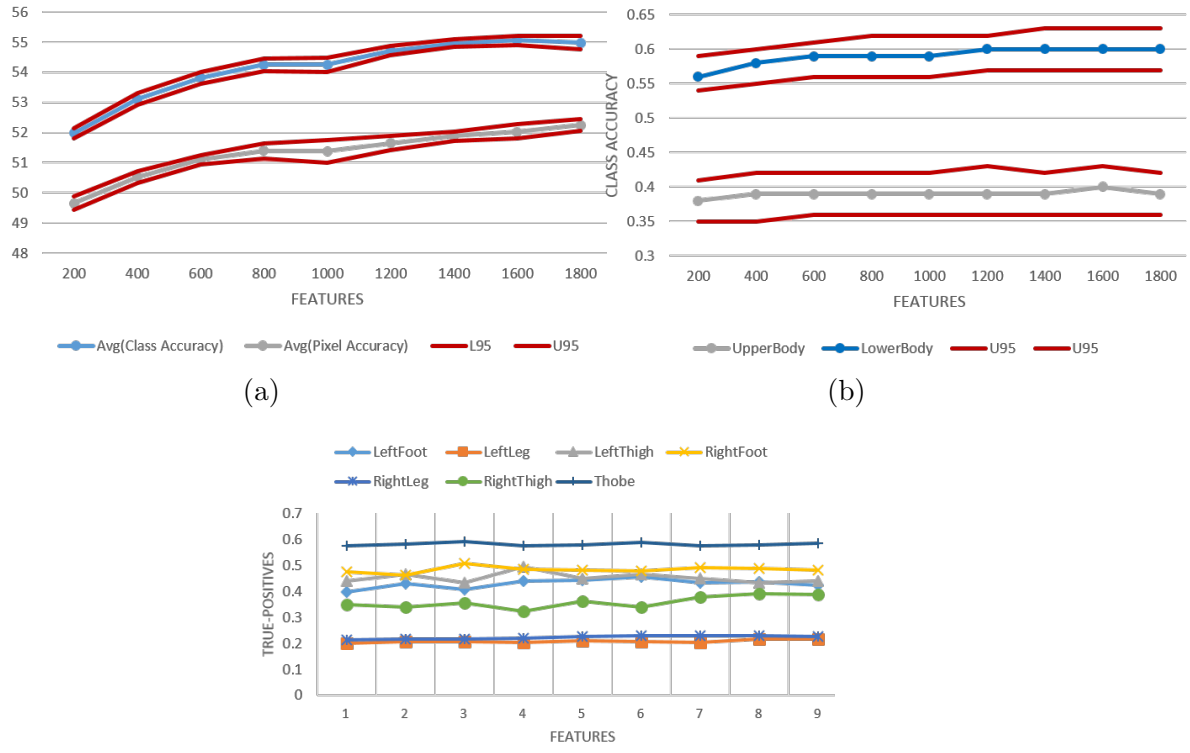


Figure 4.7: (a) average class accuracy versus number of features. (b) average class accuracy per body section (c) Average true positive for each body part in the lower section including TBP

4.5.2 Number of Features

For each node n in tree level tl we randomly sample k features $\Phi = (u,v)$ where u,v are two offsets from the sample instance x. The graph in Figure 4.7(a) shows the accuracy as the number of features increases. The graph shows an increasing gain in the accuracy by increasing number of features. Increasing the number of candidate features allows exploring wider range of possible angular and linear distance and thus increasing the chance for finding better features. Assuming two unit vectors u,v with 8 degrees (0,45,90,..315), there are total of 64 combinations

for the pair (u,v). On average, the expected number of samples before all the 64 combinations are explored is

$$\sum_{a=1}^{64} \frac{64}{a} \approx 400 \tag{4.1}$$

Figure 4.7(b) shows that higher portion of the gain comes from the upper section of the body. The lower body shows a slight improvement for greater number of features. In (c) we show the class accuracy of each part in the lower body. While shows no significant change with larger features, TBP is best estimated part among the group. Specifically, from all TBP pixels, the classifiers were able to estimate correctly 60% of them. The graph shows an inverse relationship between the LeftThigh and the RightThigh. Apparently the classifiers couldn't generalize well to distinguish left from right, and they were biased towards either one. This can be fixed in the future by further mining the training dataset and provide better examples for the classifiers.

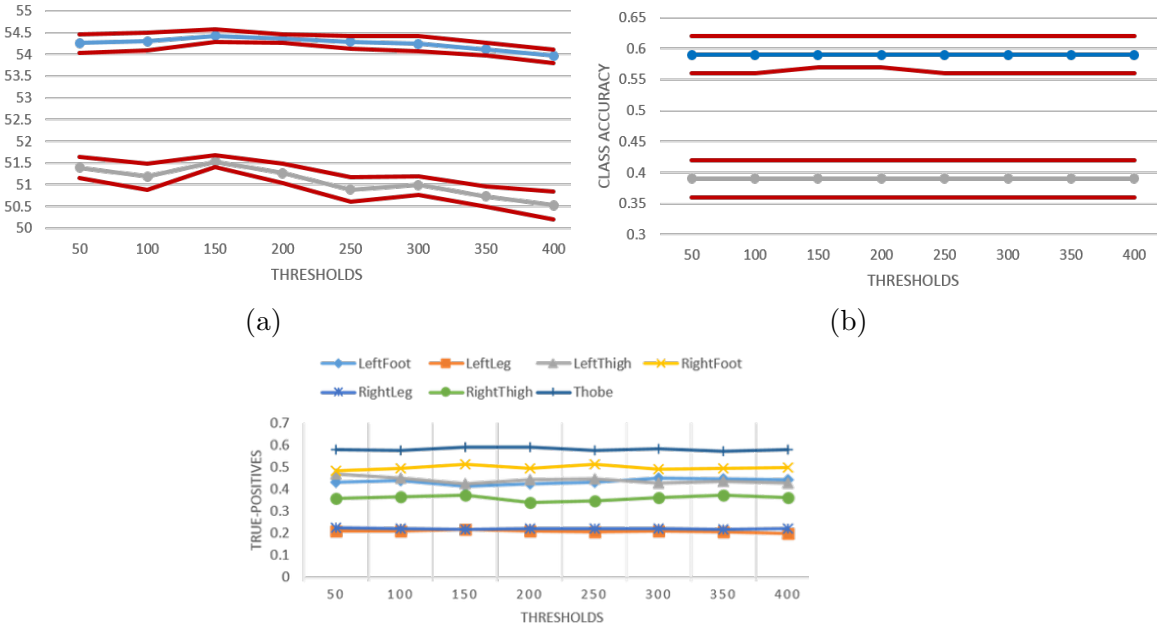
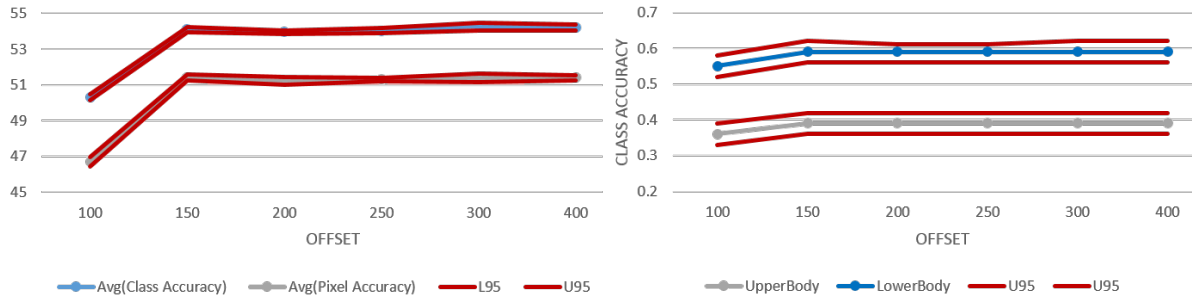


Figure 4.8: (a) average class accuracy versus number of thresholds per feature candidate. (b) average class accuracy per body section (c) average true positive for each body part in the lower section including TBP

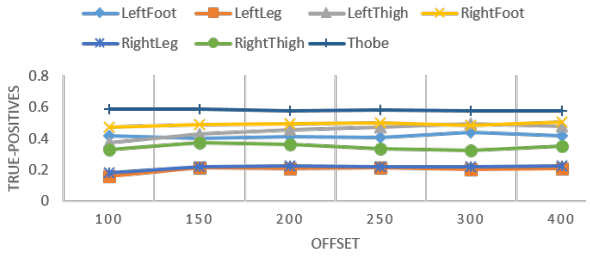
4.5.3 Number of Thresholds

For each candidate feature Φ, we sample p thresholds γ. Waldvogel implements the sampling of thresholds by randomly samples $x_1, x_2..., x_p \in D'$, and calculate there response to the candidate feature Φ. The line graph in Figure 4.8(a) shows the accuracy as the number of thresholds increases. The accuracy reaches the peak at 150 and then starts to plateau in a slightly decreasing manner. The accuracy of the upper and the lower section of the body were equally unchanged as the number of thresholds increases. The line graph in (c) demonstrates similar conclusion as the one reached with number of features regarding the lower body parts.



(a)

(b)



(c)

Figure 4.9: (a) average class accuracy versus maximum offset size. (b) average class accuracy per body section (c) average true positive for each body part in the lower section including TBP

4.5.4 Maximum Offset Size

Here, we evaluate the effect of the maximum offset size, denoted ρ, on the accuracy. While Φ's are sampled at random, they are limited by a maximum absolute value. In other words, the distance from a pixel x and an offset $o \in u, v$ should not exceed a certain limit. The graph in figure 5.14 shows how the accuracy of the classifier varies with the changing of ρ. Apparently, as the value of ρ increases, wider distance of Φ are explored and hence increasing the accuracy. The accuracy however, reaches a peak at 150 and then starts to plateau. The same result is

demonstrated in (b) for each body section. Although in (c) we see similar pattern to what have been observed in the previous two parameters, it is worth to note that the number true positives for LeftThigh increased by almost 10%. This, perhaps, indicates that classifiers tend to overfit with larger offset size.

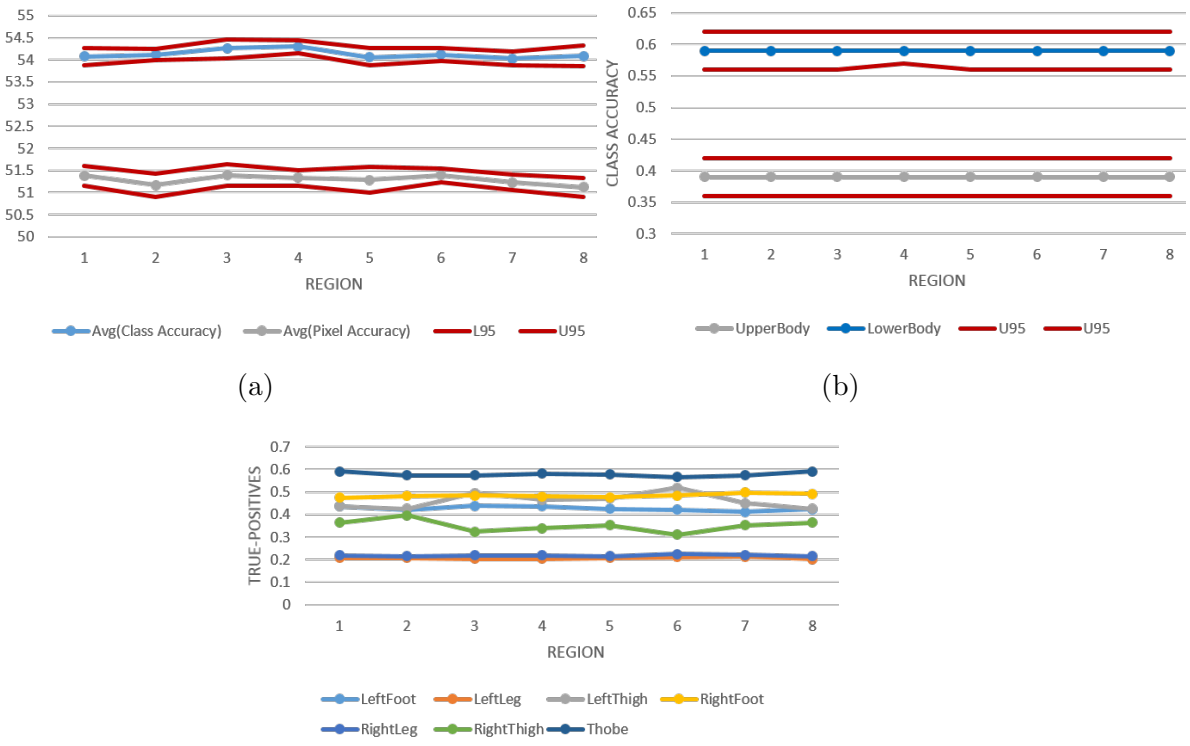


Figure 4.10: (a) average class accuracy versus maximum average region size. (b) average class accuracy per body section (c) average true positive for each body part in the lower section including TBP

4.5.5 Maximum Average Region Size

The main motivation behind Shotton features is their cheep computational cost. In this study, however, we were not limited by any computational restrictions and thus evaluated the effect of employing average-region-depth difference instead of pixel difference. We described this feature in Section 3.2.

The graph in Figure 4.10(a) shows the trend of the accuracy as the maximum region size increases. Over a range of 8 maximum region size values, the overall average class accuracy and per body section class accuracy don't show any noteworthy improvement. The only improvement noticed is associated with the

amount of true positives for LeftThigh between 1 and 6 as shown in Figure 4.10(c). However, that came, again, with an inverse cost in terms of the accuracy of RightThigh.

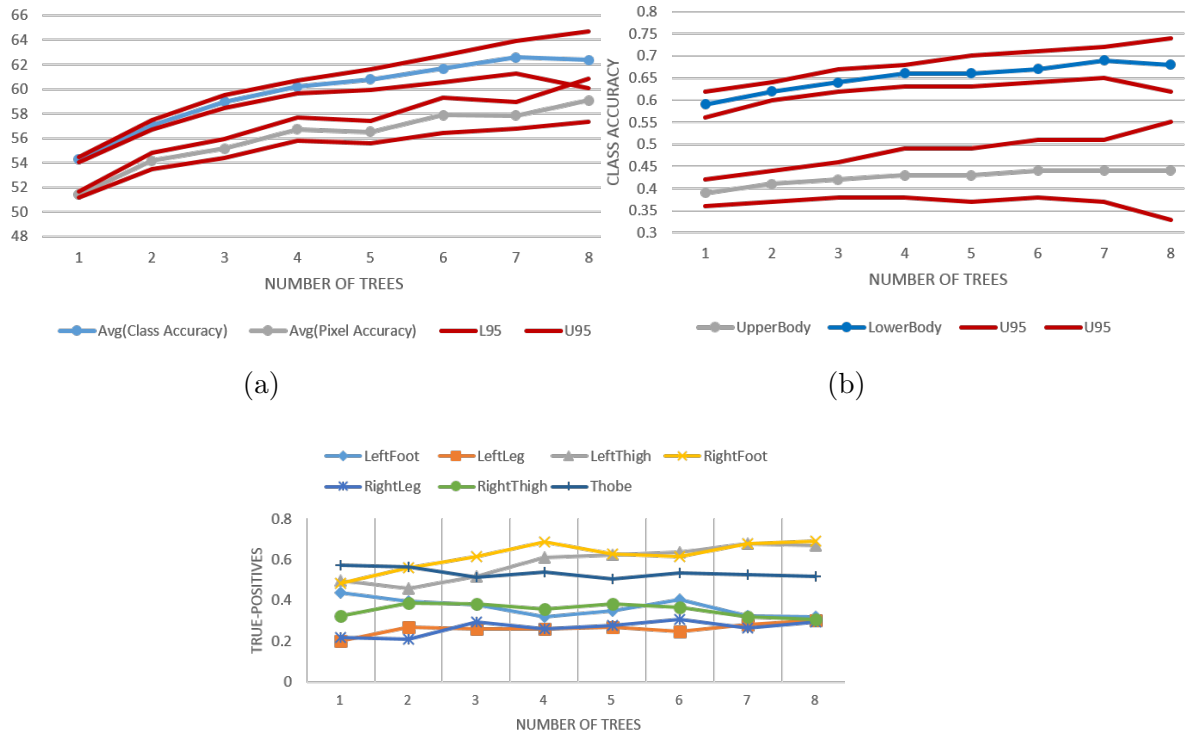


Figure 4.11: (a) average class accuracy versus number of trees in random forest classifiers. (b) average class accuracy per body section (c) average true positive for each body part in the lower section including TBP

4.5.6 Number of trees

As mentioned in chapter 3, random forest was introduced to provide an improved prediction by combining votes from multiple trees. We used the parameters values in Table 4.2 and trained 40 trees. We used these trees to create ensemble classifiers of 1-8 tree each. The average accuracy of these random forest is depicted in Figure 4.11(a).

The graph shows a clear positive influence on the accuracy as the number of trees increases to 7 trees where it then starts to saturate. The divergence in the

confidence interval from the average is due to less number of trials (e.g. 40 trees/8 trees/forest = 5 trials). The positive gain in the average accuracy is attributed to an increase in the accuracy of both the lower and the upper section of the body as can be noted from (b). Surprisingly however, the number of true positives for LeftThigh, as appears in (c), were greater than TBP for random forest with 3 or more trees. TBP true positives decreased by up to 8% for LeftThigh. This, with the other aforementioned analysis, prove a bias in the dataset toward LeftThigh, and provide a low-hanging fruit for optimization in the future. Last but not least, unlike the previous parameters, ensemble classifiers effected the estimation of the legs positively with an improvement in the number of true positives by up to 10% for each.

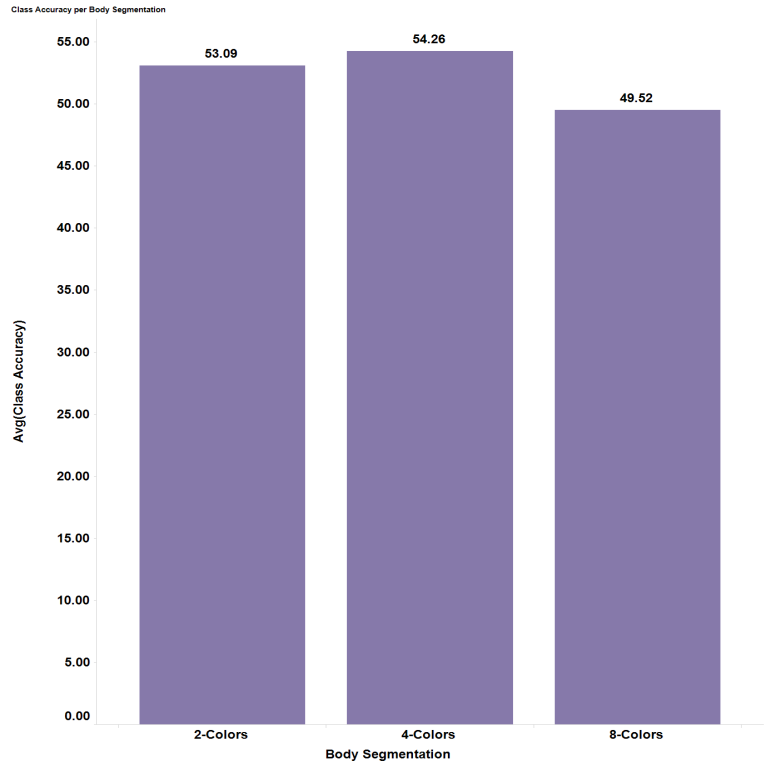


Figure 4.12: compares the average class accuracy of random decision tree classifiers trained with the 3 body segmentation: 2-colors, 4-colors, and 8-colors.

4.5.7 Body Segmentation

Earlier in this chapter, we showed 3 types of body segmentation that we used. Graph 4.12 shows a bar graph that compares the average class accuracy obtained by training 3 trees with configuration in Table 4.2 and 3 different datasets. 4-colors segmentation shows a slight increase in class accuracy over 2-colors. The graph also shows that segmenting the lower body into 8 parts has a relatively high negative impact on the class accuracy.

4.6 Effectiveness Evaluation

As shown in the previous section, the attempts to identify which configuration provides the best classification accuracy and why, were done mainly through an exhaustive search for the optimal parameters. The accuracy of each classifier were assessed using statistical measures that give a mean of how a set of predicted images compare to their respective ground truth images. Such statistical measure, however, cannot evaluate alone whether an acceptable prediction is achieved or not. In our case for example, an acceptable prediction is the one that can achieve a relatively clean segmentation of the body parts despite the occurrence of jagged edges and lines or miss-predicted pixels. In conjunction with the quantitative results above, we present a visual evaluation to explain further our outcomes.

The main purpose of conducting test on synthesized dataset is to elicit ideas, and insights about the general behavior of the trained trees that can guide further enhancements. Figure 4.13 shows a few examples from our three testing datasets predicted by our best quantitevly accurate classifier. Collectively, the output of these datasets provide a means of determining the degree of success of our experiments and what could be improved in the future. The following points were drawn from a comprehensive study of the predicted images, the confusion matrix depicted visually in Figure 4.14, the ROC curve in Figure 4.15, the precision values in Figure 4.16, recall values in Figure 4.17, and the f-score values in Figure 4.18:

- Overall, the classifiers were able to learn some features to distinguish each body part as can be deduced from Figure 4.14, Figure 4.15, and Figure 4.18.

(a)

(b)

(c)

Figure 4.13: A set of pairs of prediction and ground truth images that were sampled from our testing datasets (a) 2-colors, (b) 4-colors, and (c) 8-colors). The examples here visually summarize the behavior of the random forest that were used to generate them.

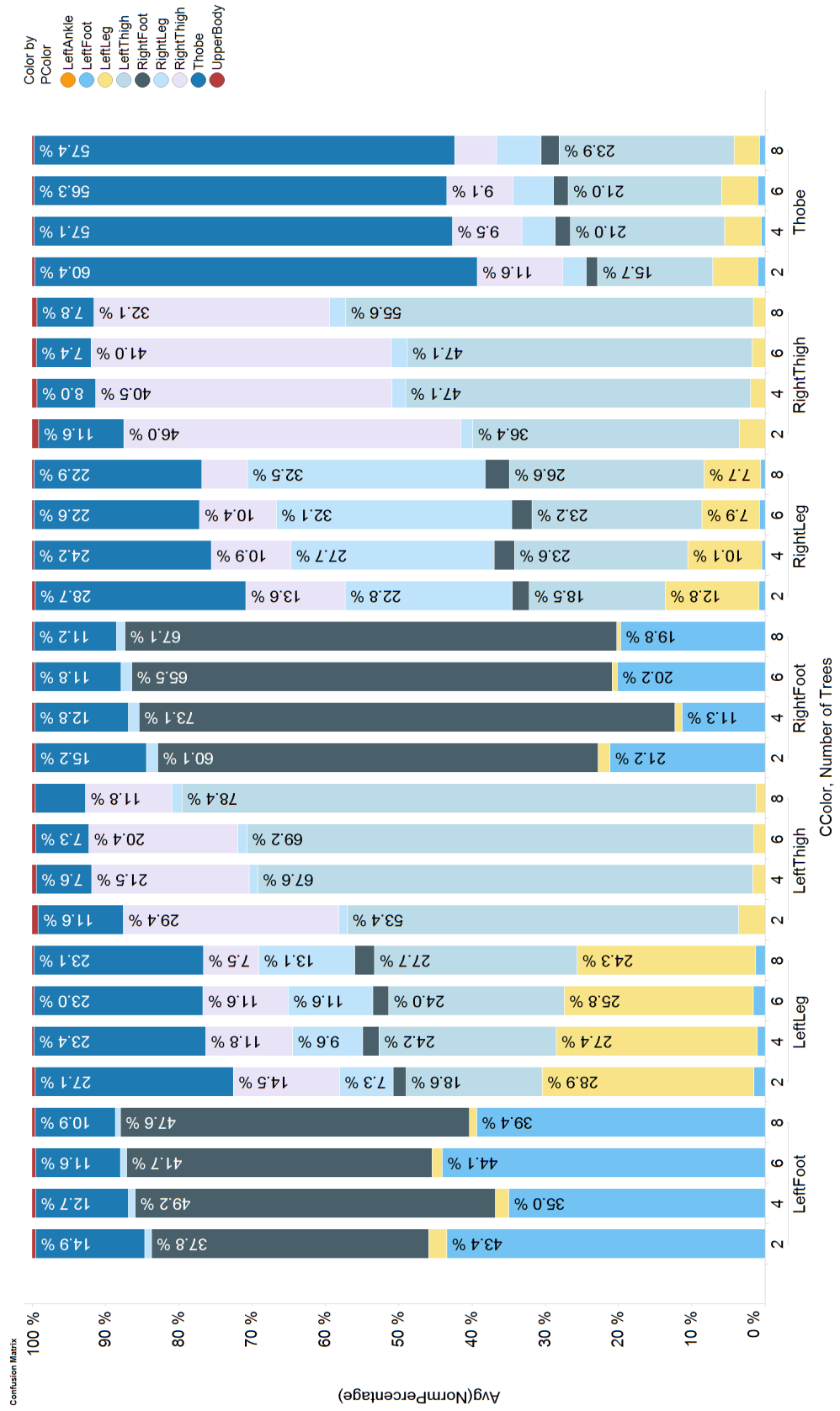


Figure 4.14: Visualization of prediction confusion matrix grouped by number of trees in ensemble classifiers

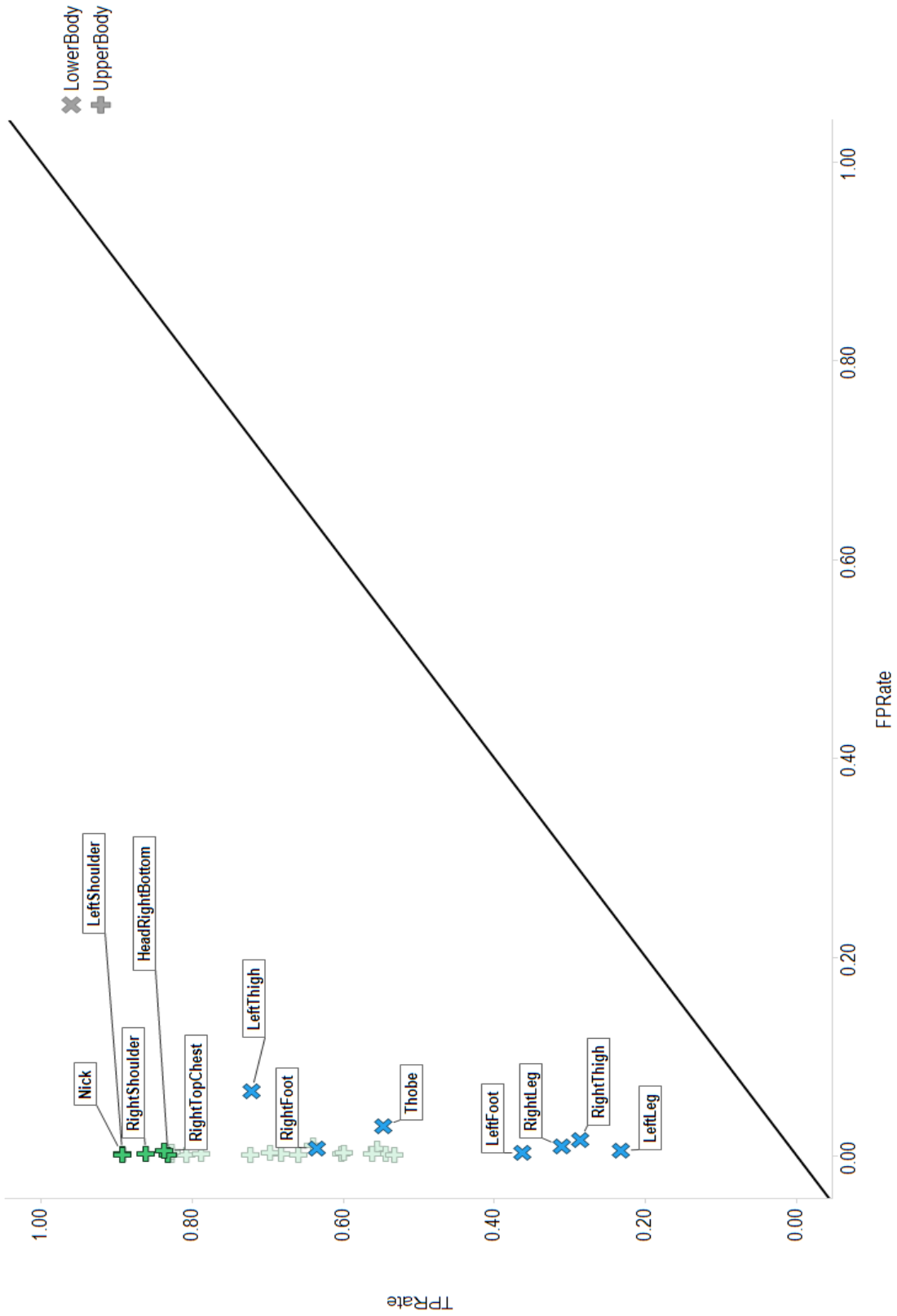
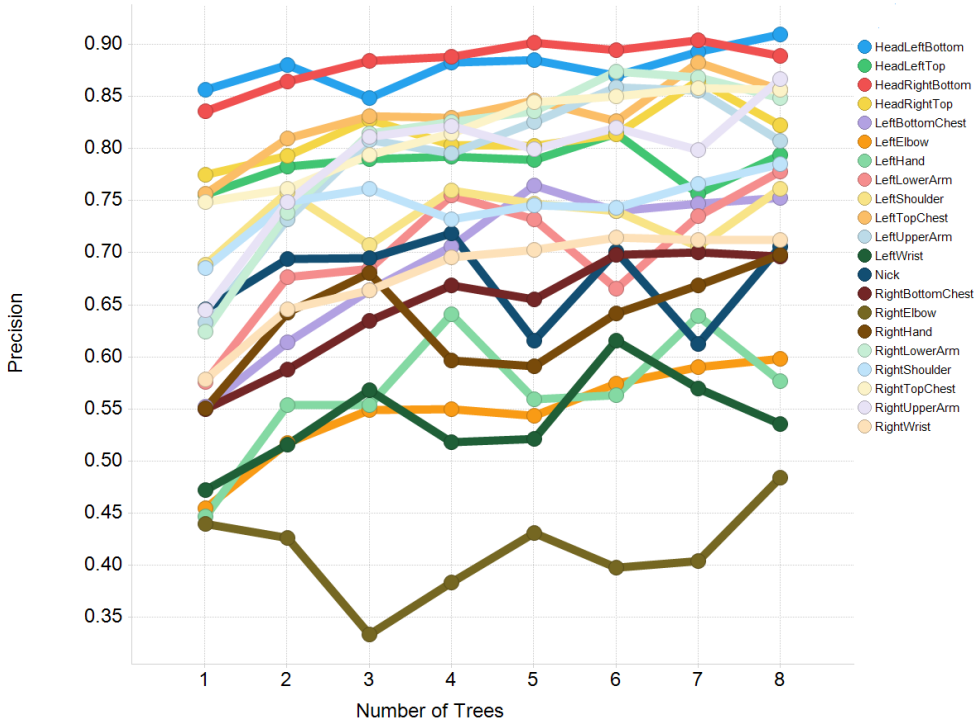
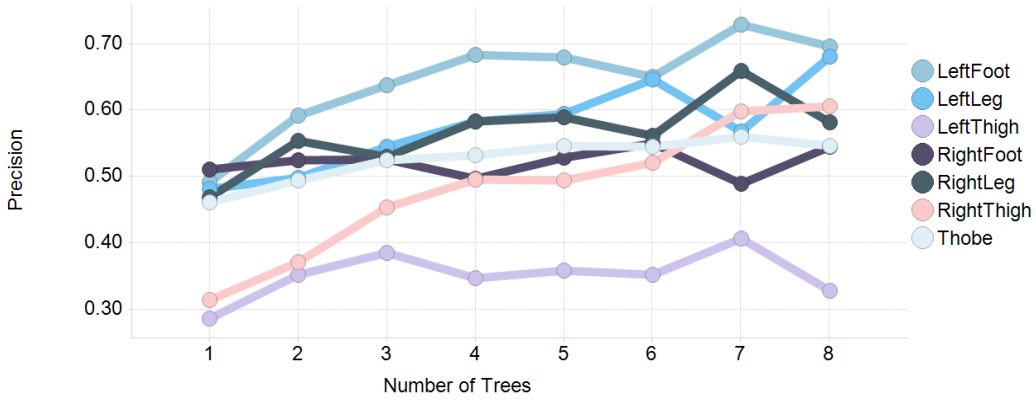


Figure 4.15: ROC curve showing the discrete performance of Thobe classifiers on each body part

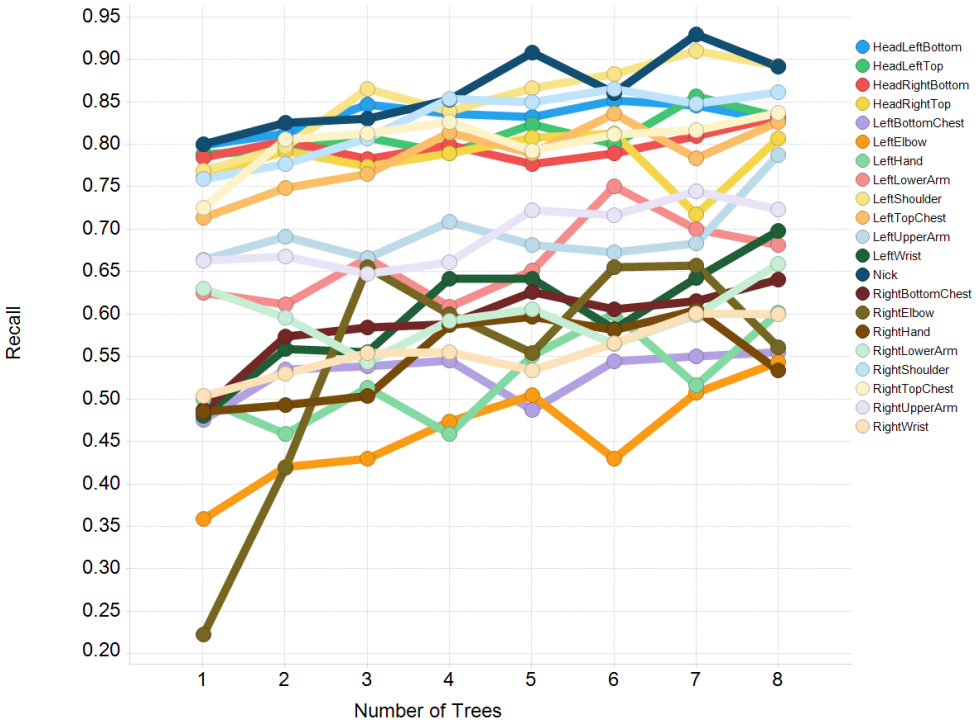


(a) Precision Upper Body

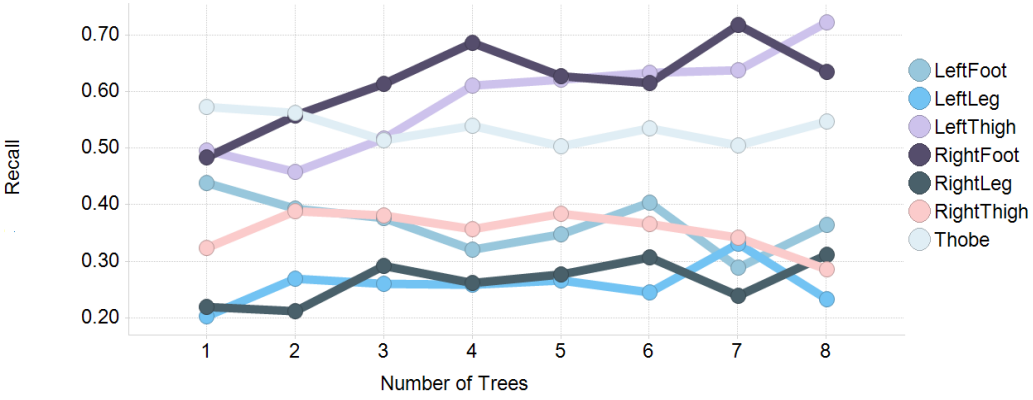


(b) Precision Lower Body

Figure 4.16: A set of figures that shows the precision values for (a) upper body section, and (b) lower body section as the number of ensembled trees increases

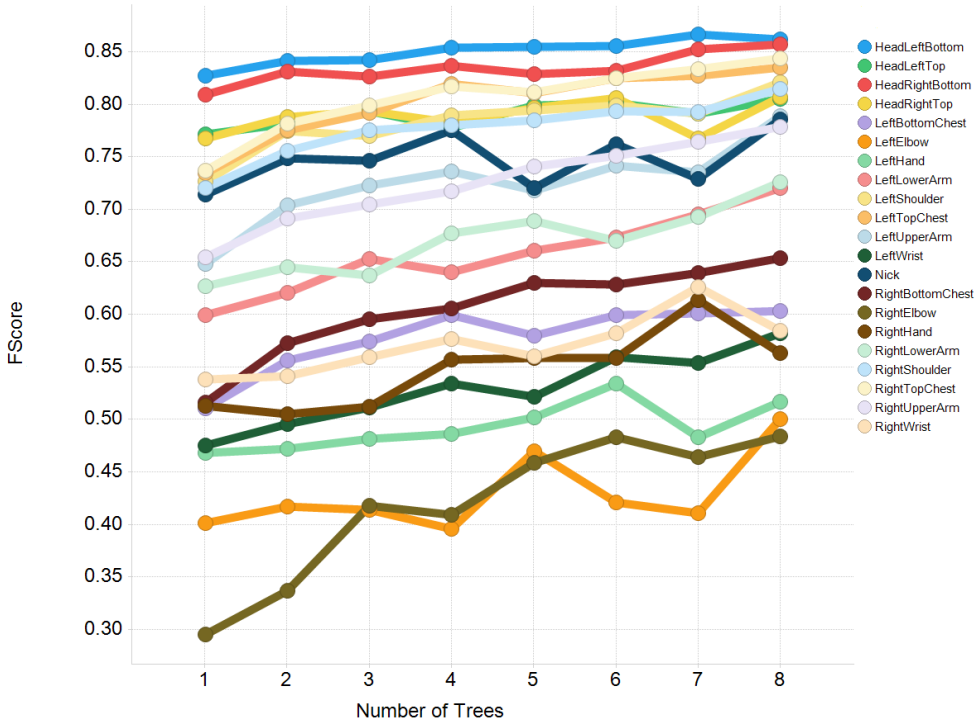


(a) Recall Upper Body

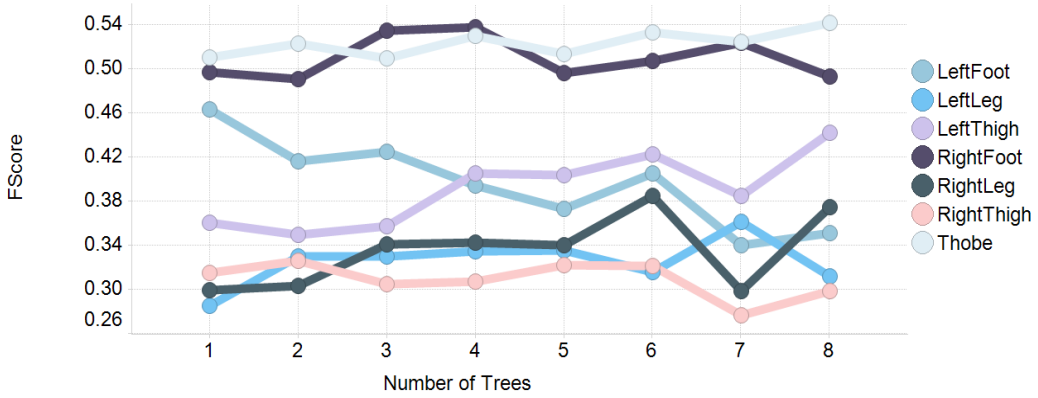


(b) Recall Lower Body

Figure 4.17: A set of figures that shows the recall values for (a) upper body section, and (b) lower body section as the number of ensembled trees increases



(a) F-Score Upper Body



(b) F-Score Lower Body

Figure 4.18: A set of figures that shows the f-score for (a) upper body section, and (b) lower body section as the number of ensembled trees increases

The classifiers, however, show different behavior with each body part. Figure 4.15 and the recall values in Figure 4.16 show that the classifiers are conservative with some body parts, making few false positive errors, but having also low true positive rate for some body parts.

- As far as the upper body is concerned, the classifiers perform quite well in segmenting it into the predefined body parts with a very high confidence in some body parts(e.g. 4 head segments, nick, and shoulders) Figure 4.15 and Figure 4.16(a)

- Figure 4.14 clearly shows that the classifiers were able to learn the topology of the body. As for the lower body, the classifier did learn to distinguish the lower body parts pixels. This can be seen in that rarely a pixel in the lower body is labeled with one of the upper body parts' colors. See the very small black areas in Figure 4.14 that represents a lower body pixel classified as upper body.

- In lower section of the body, the classifiers successfully distinguishes between left and right in some cases and fails horribly on others. Figure 4.14 demonstrates how the left thigh and the right thigh are being misclassified interchangeably. The fact that classifiers are biased towards the left side becomes clear as the number of trees increases. Figure 4.15 and the precision values in Figure 4.16(b) show that classifiers tend to select the left thigh and the right foot more often and thus score a lot of true positives for these parts (i.e. decreasing the precision) while being relatively conser-

vative in respect to the opposite parts. This could be attributed to a bias in the dataset that lead to an over-fitting in the classifiers. Although we used class uniform sampling when sampling the pixels, the variability of the poses in the training dataset might be higher for certain body part, giving it an advantage over its opposite body part. The fact that the confusion between left thigh and right thigh increases as the number of trees increases support this hypothesis. For example, since each tree is biased to the left thigh with some degree, even if some trees vote right thigh for certain pixel, the consensus vote is biased to the left thigh. It is worth to note, however, in Figure 4.14 that the classifiers mostly confuse left thigh with right thigh, and left foot with right foot but not with other body parts. A directive training could be used by an intervention at some points during the training process. In other words, instead of completely random selection of features, we can manually identify some offset features that will help to distinguish left from right.

- The TBP classifier has a consistent and higher F1-score than the other 4 lower section body parts (excluding feet) with a slow improvement as the number of trees increases Figure 4.18(f). Regardless of the accuracy of the prediction of the other lower section parts, apparently the classifier learned how to identify the pixel-area that doesn't belong to any lower body part. The precision increases slowly as the number of trees increases, indicating less false positive for the TBP, and hence becoming more conservative Fig-

ure 4.14(b) and Figure 4.16(b).

- Compared to the WCB classifier, Thobe classifier has less accuracy in distinguishing between the left and the right foot. The reason could be that the features that were used to achieve such task were faded by the presence of the dress.

4.6.1 Real Depth Images

Visual findings can be accompanied with quantitative data to give a relative measures that future or other studies can relate to. As for the real test in this study, we couldn't find a Thobe-Based dataset with ground truth information that we can use. Instead, we illuminate in the following points our observations from testing our classifier on our real dataset (see Section 4.1.2).

- From Figure 4.19 (a & b), we can see an example where, with all the noise that a real Thobe adds to the depth image, the classifier was able to achieve roughly 60% true positives.

- The classifier was able to learn the topology of the TBP (Thobe Part) which is often located at the triangular area between the legs.

- Similar to the prediction of the synthetic images, due to the disappearance of the background area between the legs, the classifier seems to lack the ability to distinguish between the left and the right foot.

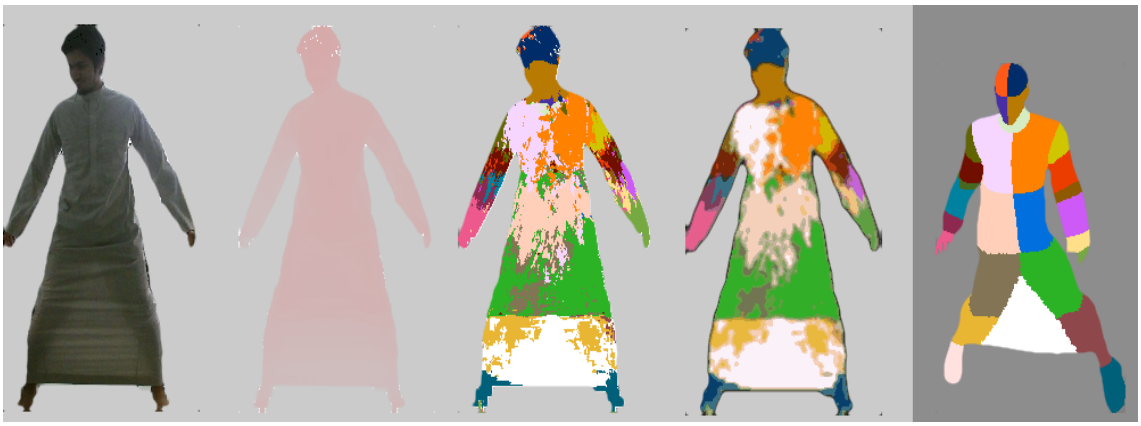
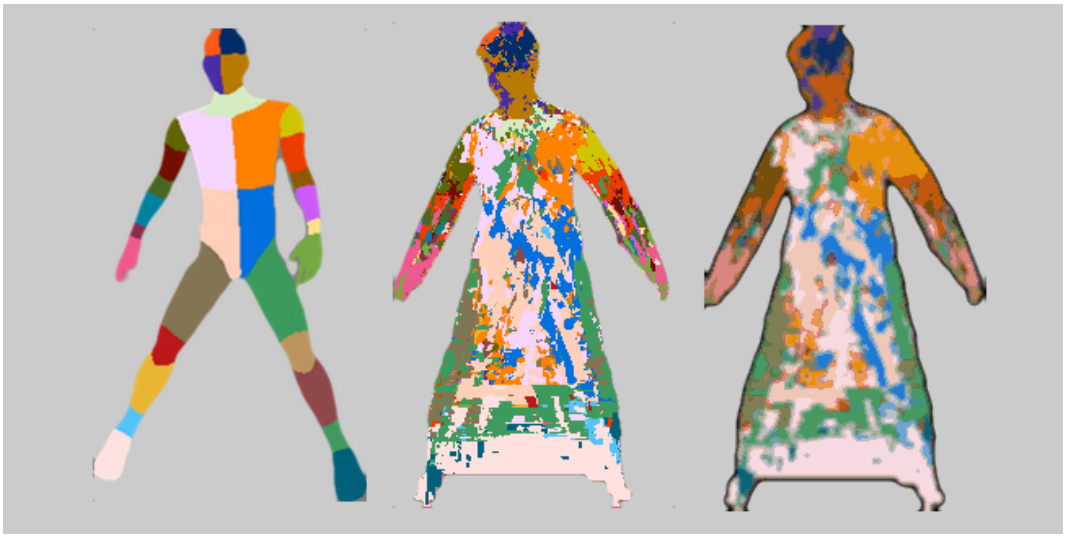


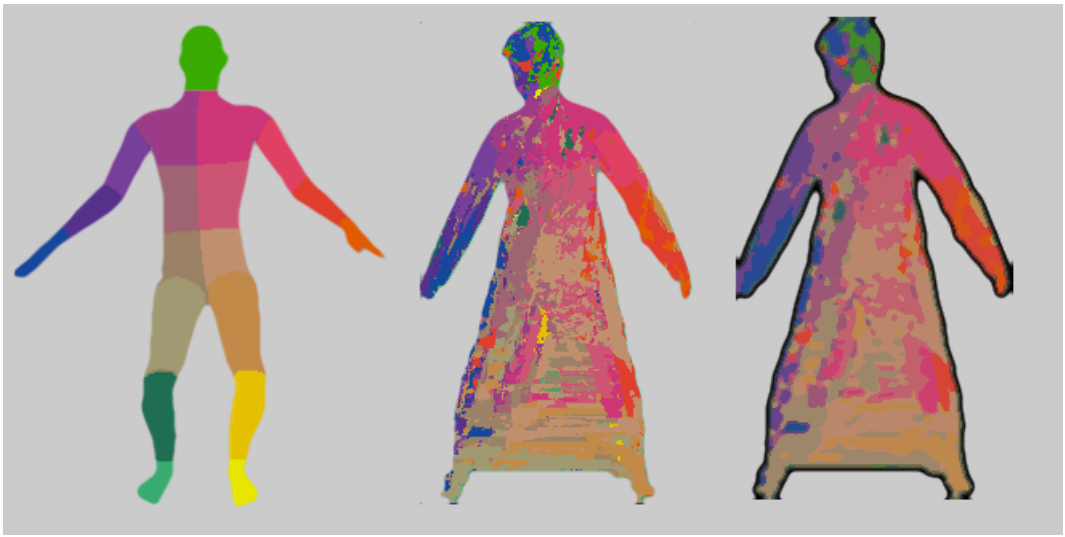
Figure 4.19: Real test of a human wearing Thobe captured by Kinect v2. From left to right: RGB image, depth image, prediction, prediction after applying k-mean segmentation, synthetic ground truth image that the pose looks similar to the real case

A classic AI system is highly tuned for a specific problem and therefore its prediction accuracy relies heavily on the degree of similarity between the training and testing conditions. The base-model used by Shotton to generate his training data do not include clothing variation similar to ours, and thus the system is not expected to perform well in such condition. However, since Shotton's data is not published and neither of Denils' and our data have similar variety and quality, we can confirm this hypothesis only by the evaluation of Kinect tracking framework as we have done earlier in this chapter. Nevertheless, it is noteworthy to show the results of testing western-cloth classifier on real human wearing Thobe. In Figure 4.20, we show the prediction of the same frame in Figure 4.19 of two classifiers: one trained with Denils' data and one trained with ours. Again, while the classification of the pixels in upper body includes many true positives, the classification of the pixel in the lower body is totally random and meaningless. Looking closely to the parts below the torso in both (a & b), one can count large

number of pixels labeled with abdomen's colors. If this was the case with Shotton prediction, then that explains why the torso joint in Figure 4.24 and 4.25 are located way below its correct location.



(a)



(b)

Figure 4.20: (a) set of pairs of prediction set of pairs of predictiontrained with our dataset (nothobe). (b) Right: Real test of a human wearing Thobe using western-cloth classifier trained with Danil's dataset. From left to right in a,b: synthetic image that shows the ground truth segmentation of similar pose, classifier prediction, prediction after applying k-mean segmentation

4.7 Threat to Validity

In this study, we focused on creating a representative model that can be used to synthesize a dataset for training decision trees to estimate the body parts of human wearing Thobe. The implementation of the other concepts discussed earlier including training random forest, and impurity score is outsourced. In this section, we discuss some aspect that might be cause a threat to the validity of our results.

Random Forest Trainer and Classifier

All the trees that we built in our research, whether it was used for the analysis above or not, were trained using the implementation in[156]. We verified that the code includes all the features required to carry out the study. However, we gave little attention to the validity of the implementation. Should newer version of the package be released, or different packages is used, then the results are prone to some errors or inconsistency.

Testing on Real Datasets

We have generated our own real dataset using Kinect v2 to test the effectiveness of our trees. This dataset, however, is not released and, as far as this study is concerned, is not meant to be a standard for comparison. Moreover, should a different version of Kinect or another environment set-up be used to create real dataset, then trees trained with our dataset have no base for performance expectation.

(a)

(b)

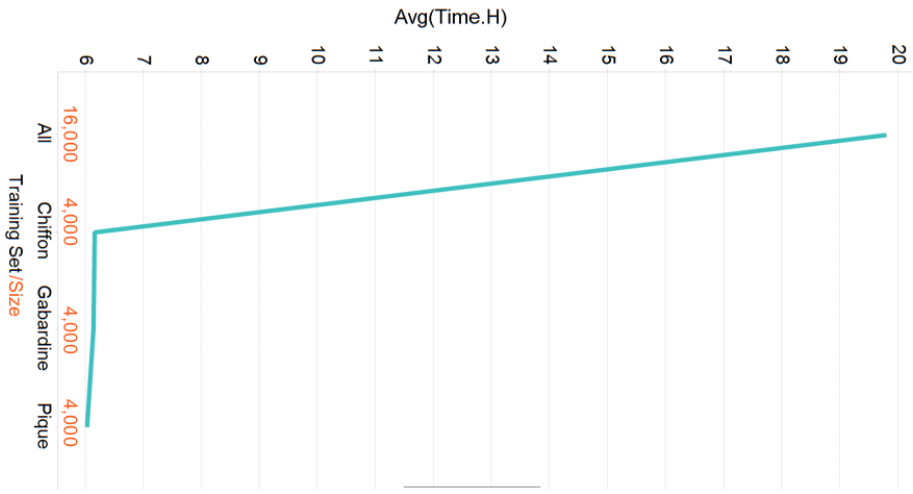
(c)

Figure 4.21: Example of labeled synthesized datasets. (a) lower body is segmented into two parts: left and right. (b) lower body is segmented into 4 parts: 2 thighs, 2 legs. (c) lower body is segmented into 8 parts: two thighs, two knees, two legs, two ankles

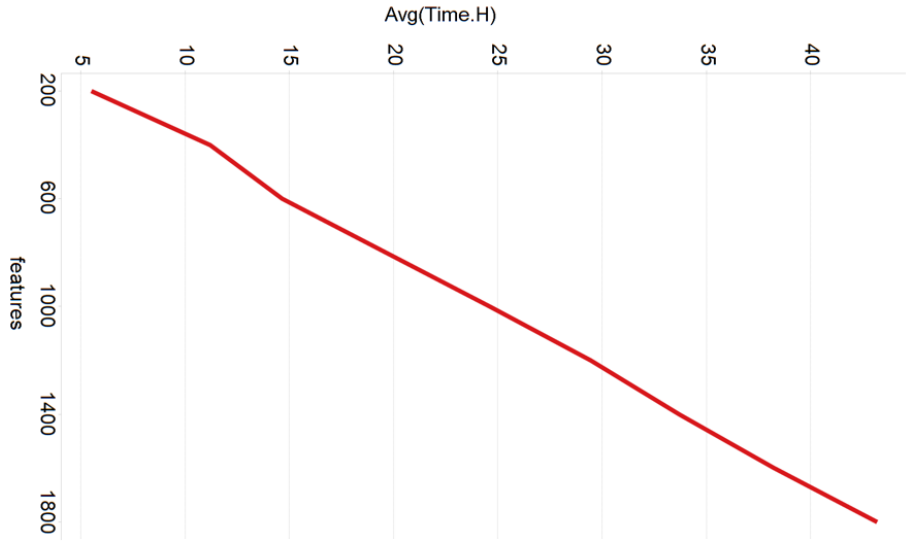
(a)

(b)

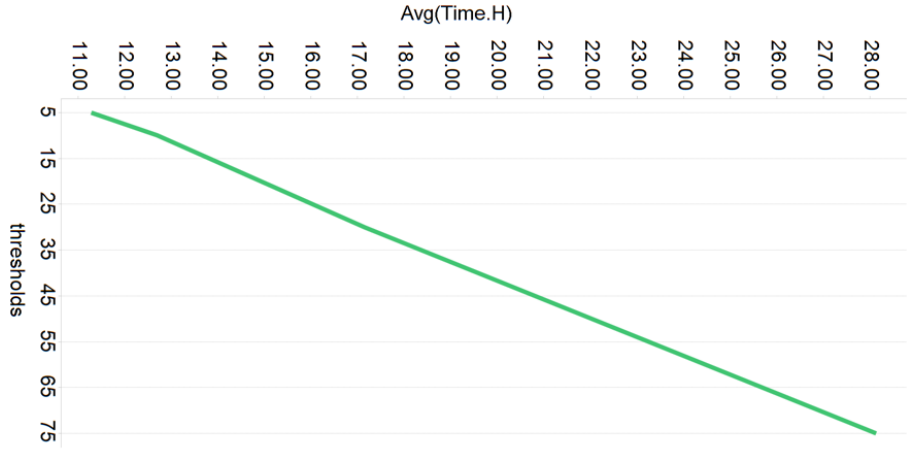
Figure 4.22: A set of RGB-D images captured using Kinect v2 (a) pants, (b) Thobe



(a)

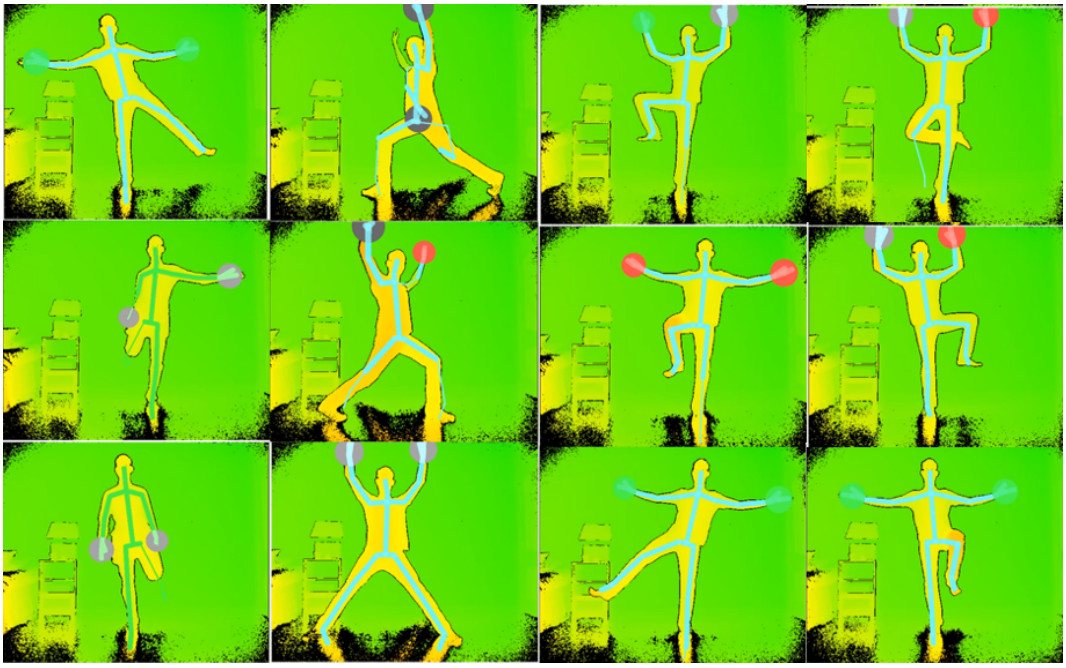


(b)

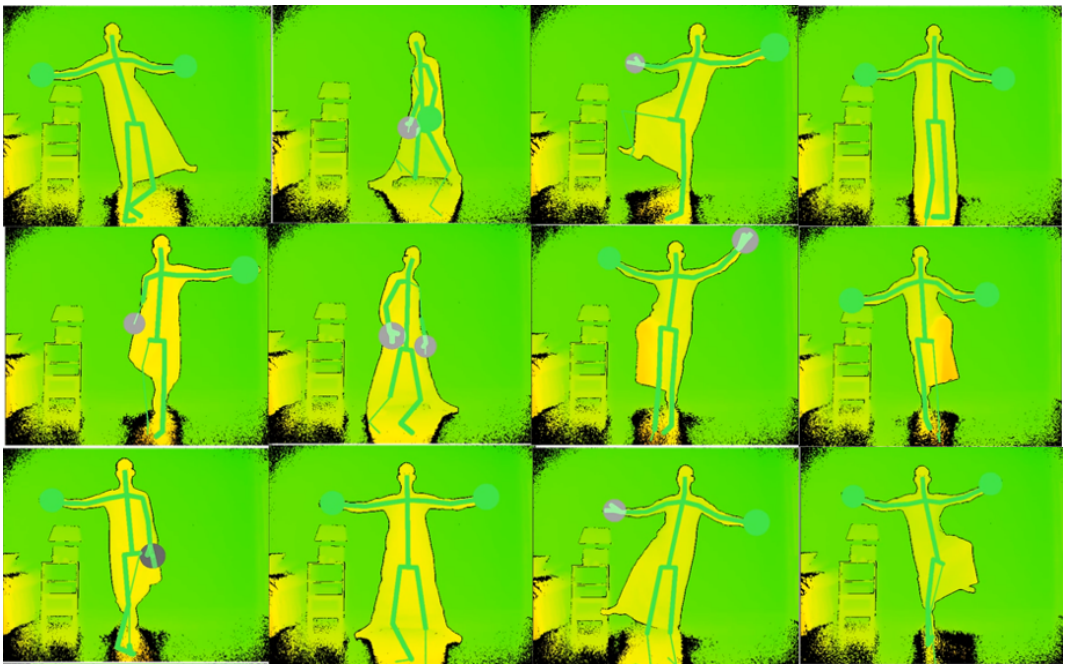


(c)

Figure 4.23: The behavior of training time of a single tree as (a) dataset size increase, (b) the number of features, and (c) thresholds



(a)



(b)

Figure 4.24: (a) Evaluation of Kinect v2 body tracking framework with western cloths (b) Evaluation with Thobe

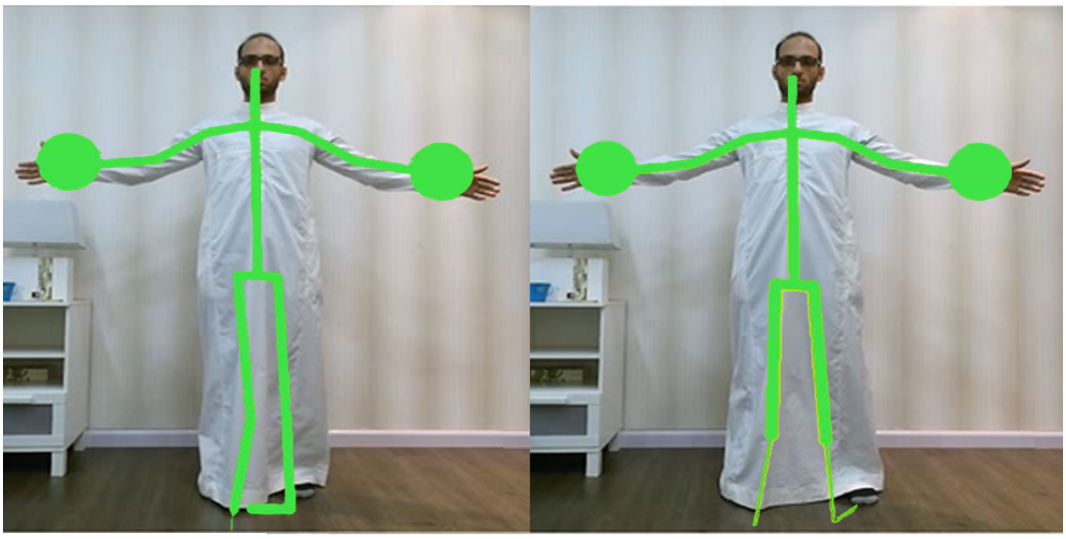


Figure 4.25: The predicted skeleton by Kinect v2 body tracking framework for two T-Poses

CHAPTER 5

CONCLUSION AND OUTLOOK

5.1 Thesis Summary and Conclusion

Human pose estimation with chaotic dress variation is a challenging problem. In fact, even human performs imperfectly if tested on static images solely. Building on the success of Shotton solution, this thesis has proposed a remedy to this problem. In this study, we addressed this problem through testing the viability of estimating the human pose of a person wearing Arabic Thobe. Inspired by Shotton's intermediate body representation, and using depth images, depth comparison features, and random forest, we showed that improvements on pose estimation with such clothing variation can be achieved.

Fundamentally, our approach was to train random decision trees to estimate human pose from depth images only. Using random depth comparison features, the random decision tree were to classify each pixel in a depth image as a background or one as one of several pre-defined body parts. Each body part is given

a distinct color, and assembled together, they form the predicted pose.

In order to generate the training set, we used data-synthesis approach. Specifically, both the training depth and label images are synthesized in a virtual environment where the character is a digital human model. Unlike normal human model, we used a human model wearing Thobe. Using garment simulation engine, we were able to mimic simple interactive scenarios where the real human character is wearing Thobe. The animated base-model along with the garment simulation allowed us to synthesize the required depth and labeled data. The depth images is calculated by measuring the Euclidean distance from 16 different points of reference (virtual cameras) to the model in each frame. In order to generate the ground truth data for the synthesized depth images, we use texture mapping for labeling the upper body section and our introduced dynamic technique for labeling the lower section. The dynamic labeling assign a Thobe pixel a body part color (thigh, knee, leg, or ankle) if it lies within a 3D threshold distance from that body part. The area of pixels that do not belongs to any body part are considered Thobe body part and labeled with white.

We examined the effectiveness of using our synthesized data to train many random decision trees and random decision forest. We found that the trained decision trees and forests were able to classify pixels in synthesized depth images into their body part with relatively good accuracy. The accuracy varied with the training configuration. Next, we examined the effectiveness of the random forest that achieved best accuracy on synthesized dataset when exposed to real human

depth images taken from Microsoft Kinect. By visually evaluating the prediction of the random forest trained with our Thobe base-model data, we found that classifiers were able to perform quite well in the upper section, and fairly good with some left-right confusion in the lower section. Comparatively, we found that random decision forests trained with normal base-model data generate completely random senseless classifications in both upper and lower section.

5.2 Outlook

Building on the success of Shotton solution, this thesis has proposed a remedy to the human pose estimation where clothes are not restricted to western style. The reported results using Shotton solution have clearly corroborated our assumptions. Following the same spirit, the presented work has also its own shortcomings and limitations. Therefore, many opportunities for extending the scope of this thesis remain. In this section, we present some of these opportunities.

Larger and more Diversified Dataset

The dataset used in this research is limited to 1000 frames of simple human activities. Widening the scope of the moves by employing mocap data, for example, as done by Shotton in [6] will help to increase the dataset size. Moreover, each image can be flipped horizontally and vertically during training to optimize the feature selection and hence improve the generalization. This, in fact, can solve the left-right issues that we faced in our experiments.

Assisted Training

The features used in this thesis are randomly sampled. However, when identifying the body parts, models generated with such features have been shown to be less effective in distinguishing between left and right body parts in the lower section. An assisted training aims to intervene and direct the training process when necessary. For example, if while training a node, the probability of a left body part is similar to its opposite right body part, a fixed feature that is tuned to such case is better to be employed.

Batch Training

The trees generated in this thesis are trained with a single large batch of random examples. The disadvantage of this approach is that nodes sample their best split feature based on the examples reached them. Batch training will allow validation of the so-far trained tree structures with another set of examples and hence, re-adjust bad nodes accordingly. Moreover, similar to neural network, where each batch is treated as single example and thus examples in a batch will adjusts weights and bias of the network only once, batch training could be used to train the leaf nodes, or the full structure of the tree.

Abstract Model

Instead of using human wearing Thobe base-models to synthesize training data, employing person-dependent model with some mechanism to estimate the pose will be another potential way to improve the generalization. In Section 1.2.3, we discussed some works that use person-dependent models. The depth images are noisy and, hence, by employing models that are defined with fixed shapes,

and body parts are set in reference to each other[86], for example, the noise will be averaged out.

REFERENCES

[1] "The history of visual communication." [Online]. Available: http://www.citrinitas.com/history_of_viscom/rockandcaves.html

[2] F. Remondino, *TOF Range-Imaging Camera*. Heidelberg New York Dordrecht Londo: Springer., 2013.

[3] K. Khoshelham, "Accuracy and resolution of kinect depth data for indoor mapping applications," *Sensors*, vol. 12, no. 2, pp. 1437 –1454, 2012.

[4] Y. S. Chang, "A kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," *Res. Dev. Disabil*, vol. 32, pp. 2566–2570, 2011.

[5] C.-C. C. L. Xia, "Human detection using depth information by kinect. computer vision and pattern recognition workshops (cvprw),," in *IEEE Computer Society Conference*. Colorado Springs, CO: IEEE, 2011, pp. 15 – 22.

[6] J. Shotton., A. Fitzgibbon., M. Cook., T. Sharp., M. Finocchio., R. Moore., A. Kipman., and A. Blake, "Real-time human pose recognition in parts from single depth images." *CVPR*, 2011.

[7] R. Klette, *Concise computer vision*. Springer, 2014.

[8] K. R. Gegenfurtner and J. Rieger, "Sensory and cognitive contributions of color to the recognition of natural scenes," *Current Biology*, vol. 10, no. 13, pp. 805–808, 2000.

[9] J. Tanaka, D. Weiskopf, and P. Williams, "The role of color in high-level vision," *Trends in cognitive sciences*, vol. 5, no. 5, pp. 211–215, 2001.

[10] S. Hagen, Q. C. Vuong, L. S. Scott, T. Curran, and J. W. Tanaka, "The role of color in expert object recognitionrh: Short title??" *Journal of vision*, vol. 14, no. 9, pp. 9–9, 2014.

[11] M. Nitsche, J. M. Turowski, A. Badoux, D. Rickenmann, T. K. Kohoutek, M. Pauli, and J. W. Kirchner, "Range imaging: a new method for high-resolution topographic measurements in small-and medium-scale field sites," *Earth Surface Processes and Landforms*, vol. 38, no. 8, pp. 810–825, 2013.

[12] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–I.

[13] A. Wittmann, A. Al-Nuaimi, E. Steinbach, and G. Schroth, "Enhanced depth estimation using a combination of structured light sensing and stereo reconstruction."

[14] Inside the intel realsense gesture camera. [Online]. Available: http://www.chipworks.com/about-chipworks/overview/blog/inside-the-intel-realsense-gesture-camera

[15] T. Cao, Z.-Y. Xiang, and J.-L. Liu, "Perception in disparity: An efficient navigation framework for autonomous vehicles with stereo cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2935–2948, 2015.

[16] J. Guo and S. Li, "Hand gesture recognition and interaction with 3d stereo camera," *The Project Report of Australian National University*, 2011.

[17] S. Ullman *et al.*, *High-level vision: Object recognition and visual cognition*. MIT press Cambridge, MA, 1996, vol. 2.

[18] [Online]. Available: http://image-net.org/challenges/LSVRC/2016/results

[19] A. Hoogs, J. Rittscher, G. Stein, and J. Schmiederer, "Video content annotation using visual analysis and a large semantic knowledgebase," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2. IEEE, 2003, pp. II–II.

[20] Y. Hirano, C. Garcia, R. Sukthankar, and A. Hoogs, "Industry and object recognition: Applications, applied research and challenges," in *Toward Category-Level Object Recognition*. Springer, 2006, pp. 49–64.

[21] J. Aggarwal, "Motion analysis: Past, present and future," in *Distributed Video Sensor Networks*. Springer, 2011, pp. 27–39.

[22] C. Lassner, "An analysis of successful approaches to human pose estimation," Master's thesis, Universität Augsburg, 2012.

[23] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1239–1258, 2010.

[24] D. Patil, M. R. Khanderay, and T. Padvi, "Survey on moving body detection in video surveillance system."

[25] H. S. Parekh, D. G. Thakore, and U. K. Jaliya, "A survey on object detection and tracking methods," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 2, pp. 2970–2979, 2014.

[26] N. A. Ogale, "A survey of techniques for human detection from video," *Survey, University of Maryland*, vol. 125, no. 133, p. 19, 2006.

[27] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[28] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[29] M. Camplani, A. Paiement, M. Mirmehdi, D. Damen, S. Hannuna, T. Burghardt, and L. Tao, "Multiple human tracking in rgb-d data: A survey," *arXiv preprint arXiv:1606.04450*, 2016.

[30] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.

[31] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[32] D. M. Gavrila, "A bayesian, exemplar-based approach to hierarchical shape matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1408–1421, 2007.

[33] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 90–97.

[34] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[35] D. T. Nguyen, W. Li, and P. O. Ogunbona, "Human detection from images and videos: a survey," *Pattern Recognition*, vol. 51, pp. 148–175, 2016.

[36] D. M. Gavrila and V. Philomin, "Real-time object detection for" smart" vehicles," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 1. IEEE, 1999, pp. 87–93.

[37] T. Teixeira, G. Dublon, and A. Savvides, "A survey of human-sensing: Methods for detecting presence, count, location, track, and identity," *ACM Computing Surveys*, vol. 5, no. 1, 2010.

[38] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[39] A. W. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[40] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Aerospace/Defense Sensing, Simulation, and Controls*. International Society for Optics and Photonics, 2001, pp. 95–102.

[41] X. Zhou, W. Hu, Y. Chen, and W. Hu, "Markov random field modeled level sets method for object tracking with moving cameras," in *Asian Conference on Computer Vision*. Springer, 2007, pp. 832–842.

[42] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[43] M. Godec, P. M. Roth, and H. Bischof, "Hough-based tracking of non-rigid objects," *Computer Vision and Image Understanding*, vol. 117, no. 10, pp. 1245–1256, 2013.

[44] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 983–990.

[45] X. Mei and H. Ling, "Robust visual tracking using l1 minimization," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 1436–1443.

[46] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[47] Z. Qigui *et al.*, "Search on automatic target tracking based on ptz system," in *Image Analysis and Signal Processing (IASP), 2011 International Conference on.* IEEE, 2011, pp. 192–195.

[48] C. Yang, R. Duraiswami, and L. Davis, "Fast multiple object tracking via a hierarchical particle filter," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 1. IEEE, 2005, pp. 212–219.

[49] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 142–149.

[50] H. T. Nguyen and A. W. Smeulders, "Robust tracking using foreground-background texture discrimination," *International Journal of Computer Vision*, vol. 69, no. 3, pp. 277–293, 2006.

[51] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 11, pp. 1473–1488, 2008.

[52] R. Poppe, "A survey on vision-based human action recognition," *Image and vision computing*, vol. 28, no. 6, pp. 976–990, 2010.

[53] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer vision and image understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[54] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[55] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *Computer Vision and Image Understanding*, vol. 117, no. 6, pp. 633–659, 2013.

[56] L. Shao, X. Zhen, D. Tao, and X. Li, "Spatio-temporal laplacian pyramid coding for action recognition," *IEEE Transactions on Cybernetics*, vol. 44, no. 6, pp. 817–827, 2014.

[57] G. Cheng, Y. Wan, A. N. Saudagar, K. Namuduri, and B. P. Buckles, "Advances in human action recognition: A survey," *arXiv preprint arXiv:1501.05964*, 2015.

[58] F. Zhu, L. Shao, J. Xie, and Y. Fang, "From handcrafted to learned representations for human action recognition: a survey," *Image and Vision Computing*, vol. 55, pp. 42–52, 2016.

[59] A. F. Bobick and J. W. Davis, "The recognition of human movement using temporal templates," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 3, pp. 257–267, 2001.

[60] L. Wang and D. Suter, "Informative shape representations for human action recognition," in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 2. IEEE, 2006, pp. 1266–1269.

[61] N. Ikizler and P. Duygulu, "Histogram of oriented rectangles: A new pose descriptor for human action recognition," *Image and Vision Computing*, vol. 27, no. 10, pp. 1515–1526, 2009.

[62] R. Messing, C. Pal, and H. Kautz, "Activity recognition using the velocity histories of tracked keypoints," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 104–111.

[63] I. Laptev, "On space-time interest points," *International journal of computer vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[64] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*. IEEE, 2005, pp. 65–72.

[65] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1395–1402.

[66] H. Qian, Y. Mao, W. Xiang, and Z. Wang, "Recognition of human activities using svm multi-class classifier," *Pattern Recognition Letters*, vol. 31, no. 2, pp. 100–111, 2010.

[67] A. Veeraraghavan, R. Chellappa, and A. K. Roy-Chowdhury, "The function space of an activity," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 959–968.

[68] J. Yamato, J. Ohya, and K. Ishii, "Recognizing human action in time-sequential images using hidden markov model," in *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on.* IEEE, 1992, pp. 379–385.

[69] F. Negin and F. Bremond, "Human action recognition in videos: A survey," 2016.

[70] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 11, pp. 2287–2301, 2011.

[71] H. M. Dee, A. G. Cohn, and D. C. Hogg, "Building semantic scene models from unconstrained video," *Computer Vision and Image Understanding*, vol. 116, no. 3, pp. 446–456, 2012.

[72] T. B. Moeslund, A. Hilton, and V. Krüger, "A survey of advances in vision-based human motion capture and analysis," *Computer vision and image understanding*, vol. 104, no. 2, pp. 90–126, 2006.

[73] N. Sarafianos, B. Boteanu, B. Ionescu, and I. A. Kakadiaris, "3d human pose estimation: A review of the literature and analysis of covariates," *Computer Vision and Image Understanding*, vol. 152, pp. 1–20, 2016.

[74] L. Chen, H. Wei, and J. Ferryman, "A survey of human motion analysis using depth imagery," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 1995–2006, 2013.

[75] E. C. A. Xavier, Perez-Sala Sergio and G. Jordi, "A survey on model based approaches for 2d and 3d visual human pose recovery," *Sensors*, no. 14, pp. 4189–4210, 2014.

[76] T. Horprasert, D. Harwood, and L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection," in *IEEE ICCV*, vol. 99, 1999, pp. 1–19.

[77] M. Karaman, L. Goldmann, D. Yu, and T. Sikora, "Comparison of static background segmentation methods," in *Visual Communications and Image Processing 2005*. International Society for Optics and Photonics, 2005, pp. 596 069–596 069.

[78] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2. IEEE, 1999.

[79] P. D. Z. Varcheie, M. Sills-Lavoie, and G.-A. Bilodeau, "A multiscale region-based motion detection and background subtraction algorithm," *Sensors*, vol. 10, no. 2, pp. 1041–1061, 2010.

[80] A.-T. Nghiem and F. Bremond, "Background subtraction in people detection framework for rgb-d cameras," in *Advanced Video and Signal Based*

Surveillance (AVSS), 2014 11th IEEE International Conference on. IEEE, 2014, pp. 241–246.

[81] I. Schiller and R. Koch, "Improved video segmentation by adaptive combination of depth keying and mixture-of-gaussians," in *Image Analysis*. Springer, 2011, pp. 59–68.

[82] Fernandez-Sanchez., . E. J., Diaz J., and E. Ros, "Background subtraction based on color and depth using active sensors." *Sensors*, no. 7, pp. 8895–8915., 2013.

[83] K. Greff, A. Brandão, S. Krauß, D. Stricker, and E. Clua, "A comparison between background subtraction algorithms using a consumer depth camera." in *VISAPP (1)*, 2012, pp. 431–436.

[84] A. Sobral and A. Vacavant, "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos," *Computer Vision and Image Understanding*, vol. 122, pp. 4–21, 2014.

[85] K. Cannons, "A review of visual tracking," York University, Departmentof Computer Science and Engineering., Tech. Rep., (2008-07). [Online]. Available: Cannons,K.(2008-07).Areviewofvisualtracking.TechnicalReport. YorkUniversity,DepartmentofComputerScienceandEngineering.

[86] M. Siddiqui and G. Medioni., "Human pose estimation from a single view point, real-time range sensor." *CVPR*, pp. 1–2, 2010.

[87] Y. Huang and T. S. Huang, "Model-based human body tracking," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 1. IEEE, 2002, pp. 552–555.

[88] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real time motion capture using a single time-of-flight camera," *CVPR*, pp. 1, 5, 7, 8, 2010.

[89] R. Z.-L. Hu, "Vision-based observation models for lower limb 3d tracking with a moving platform." *Waterloo*, 2011.

[90] L. Bourdev. and J. Malik, "Poselets: body part detectors training using 3d human pose annotations," in *Vis*, 2009, pp. 1365–1372.

[91] L. Bourdev., S. Maji., T. Brox., and J. Malik, "Detecting people using mutually consistent poselet activations," in *Vis*, 2010, pp. 168–181.

[92] Y. Wang. and D. T. Z. Liao, "Learning hierarchical poselets for human parsing," in *Vis*, 2011, pp. 1705–1712.

[93] B. Andreas, M. Herrmann, M. Hoernig, and B. Radig, "Human body part classification in monocular soccer images." *OGRW2014*, p. 128, 2014.

[94] R. Poppe, "Vision-based human motion analysis: An overview." *Comput. Vis. Image Underst*, vol. 108, pp. 4–18, 2007.

[95] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E.-h. Zahzah, "Human pose estimation from monocular images: A comprehensive survey," *Sensors*, vol. 16, no. 12, p. 1966, 2016.

[96] D. Gavrila, "The visual analysis of human movement: A survey." *Comput. Vis. Image Underst*, vol. 73, pp. 82–98, 1999.

[97] X. Perez-Sala, S. Escalera, C. Angulo, and J. Gonzalez, "A survey on model based approaches for 2d and 3d visual human pose recovery." *Sensors 14*, vol. 3, pp. 4189–4210, 2014.

[98] H. . F. J. Chen, L.; Wei, "A survey of human motion analysis using depth imagery," *Pattern Recognit. Lett.,*, 2013.

[99] A. Agarwal and B. Triggs, "3d human pose from silhouettes by relevance vector regression," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–882.

[100] S. Gaglio, G. L. Re, and M. Morana, "Human activity recognition process using 3-d posture data," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 5, pp. 586–597, 2015.

[101] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon, "Efficient regression of general-activity human poses from depth images," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 415–422.

[102] B. Holt, E.-J. Ong, H. Cooper, and R. Bowden, "Putting the pieces together: Connected poselets for human pose estimation," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1196–1201.

[103] H. Shum, E. Ho, Y. Jiang, and S. Takagi, "Real-time posture reconstruction for microsoft kinect," *Cybernetics, IEEE Transactions on*, vol. 43, no. 5, pp. 1357–1369, Oct 2013.

[104] T. Ringbeck, "Ha 3-d time of flight camera for object detection," in *ETH Zurich, Switzerland*. Colorado Springs, CO: Optical 3-D Meas Tech, 2007.

[105] L. Li, *Time-of-Flight Camera. An Introduction Technical White Paper*, sloa190b. january 2014 revised may 2014 ed. TexasInstruments, 2014.

[106] B. Freedman, A. Shpunt, M. Machline, and Y. Arieli, "Depth mapping using projected patterns." U.S. Patent, Tech. Rep. 2010/0118123, 3 May 2010.

[107] A. L. Y. James J. Clark, *Data Fusion for Sensory Information Processing Systems*. Springer.

[108] S. Escalera, "Human behavior analysis from depth maps," in *Articulated Motion and Deformable Objects*, S. B. Heidelberg, Ed., 2012, pp. 282–292.

[109] M. Ye, Q. Zhang, L. Wang, J. Zhu, R. Yang, and J. Gall, "A survey on human motion analysis from depth data," in *In Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*, S. B. Heidelberg, Ed., 2013, pp. 149–187.

[110] C. R. J. Chen and N. Kehtarnavaz., "A survey of depth and inertial sensor fusion for human action recognition," in *Multimedia Tools and Applications*, 2015, pp. 1–21.

[111] A. J. Smit, J. M. Smit, G. J. Botterblom, and D. J. Mulder, "Skin autofluorescence based decision tree in detection of impaired glucose tolerance and diabetes," *PLoS ONE*, vol. 8, no. 6, pp. 1–7, 06 2013. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0065592

[112] (2016, 02) Master thesis: Structured learning with random forests. [Online]. Available: http://campar.in.tum.de/Students/DaStructureLearningRF

[113] T. K. Ho., "Random decision forests," in *Proceedings of the Third International Conference on Document Analysis and Recognition (ICDAR). IEEE.* IEEE Montreal, 1995, pp. 278–282.

[114] W. L. Buntine, "Decision tree induction systems: a bayesian analysis," *arXiv preprint arXiv:1304.2732*, 2013.

[115] B. Waldvogel, "Accelerating random forests on cpus and gpus for object-class image segmentation," 2013.

[116] R. L and M. O., *Decision Trees.* Springer, 2005, no. P174.

[117] S. Nowozin, "Improved information gain estimates for decision tree induction," *arXiv preprint arXiv:1206.4620*, 2012.

[118] V. Y. Kulkarni and P. K. Sinha, "Random forest classifiers: a survey and future research directions," *International Journal of Advanced Computing*, vol. 36, no. 1, pp. 1144–1153, 2013.

[119] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[120] R. C. Barros, M. P. Basgalupp, A. C. De Carvalho, and A. A. Freitas, "A survey of evolutionary algorithms for decision-tree induction," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 42, no. 3, pp. 291–312, 2012.

[121] X.-W. Chen and M. Liu, "Prediction of protein–protein interactions using random decision forest framework," *Bioinformatics*, vol. 21, no. 24, pp. 4394–4400, 2005.

[122] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[123] F. Esposito, D. Malerba, G. Semeraro, and J. A. Kay, "A comparative analysis of methods for pruning decision trees," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 5, pp. 476–491, 1997.

[124] J. J. Oliver and D. J. Hand, "On pruning and averaging decision trees," in *Machine Learning: Proceedings of the Twelfth International Conference*, 2014, pp. 430–437.

[125] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[126] S. Ren, X. Cao, Y. Wei, and J. Sun, "Global refinement of random forest," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 723–730.

[127] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[128] G. Fumera and F. Roli, "A theoretical and experimental analysis of linear combiners for multiple classifier systems," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 6, pp. 942–956, 2005.

[129] F. Ferraro, N. Mostafazadeh, T.-H. K. Huang, L. Vanderwende, J. Devlin, M. Galley, and M. Mitchell, "A survey of current datasets for vision and language research," in *Conference on Empirical Methods in Natural Language Processing*, 2015.

[130] (2016, 02) Autodesk maya. [Online]. Available: http://www.autodesk.com/products/maya/overview

[131] H. N. Ng and R. L. Grimsdale, "Computer graphics techniques for modeling cloth," *Computer Graphics and Applications, IEEE*, vol. 16, no. 5, pp. 28–41, 1996.

[132] R. Parent, *Computer animation: algorithms and techniques*. Newnes, 2012.

[133] (2016, 02) Autodesk maya. [Online]. Available: http://www.marvelousdesigner.com

[134] Y.-J. Liu, D.-L. Zhang, and M. M.-F. Yuen, "A survey on cad methods in 3d garment design," *Computers in Industry*, vol. 61, no. 6, pp. 576–593, 2010.

[135] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 775–781.

[136] F. Schroff, A. Criminisi, and A. Zisserman, "Object class segmentation using random forests." in *BMVC*, 2008, pp. 1–10.

[137] J. Stückler, N. Biresev, and S. Behnke, "Semantic mapping using object-class segmentation of rgb-d images," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3005–3010.

[138] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[139] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, "Labelme: a database and web-based tool for image annotation," *International journal of computer vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[140] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2879–2886.

[141] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *Computer Vision–ECCV 2012*. Springer, 2012, pp. 746–760.

[142] S. Johnson and M. Everingham, "Clustered pose and nonlinear appearance models for human pose estimation," in *Proceedings of the British Machine Vision Conference*, 2010, doi:10.5244/C.24.12.

[143] (2016, 02) Cornell activity datasets. [Online]. Available: http://pr.cs.cornell.edu/humanactivities/data.php

[144] (2016, 02) Princeton tracking benchmark. [Online]. Available: http://tracking.cs.princeton.edu/dataset.html

[145] G. Yu, Z. Liu, and J. Yuan, "Discriminative orderlet mining for real-time recognition of human-object interaction," in *Computer Vision–ACCV 2014*. Springer, 2014, pp. 50–65.

[146] S. T. D. K. Varun Ganapathi, Christian Plagemann, "Real time motion capture using a single time-of-flight camera," in *CVPR*, 2010.

[147] M. Denil, D. Matheson, and N. De Freitas, "Consistency of online random forests," *arXiv preprint arXiv:1302.4853*, 2013.

[148] (2016) Synthesized thobe dataset. [Online]. Available: http://www.ccse.kfupm.edu.sa/pe-bp-thobe/

[149] (2016) Kinect studio. [Online]. Available: https://msdn.microsoft.com/en-us/library/dn785306.aspx

[150] C. De Vleeschouwer, A. Legrand, L. Jacques, and M. Hebert, "Mitigating memory requirements for random trees/ferns," in *Image Processing (ICIP), 2015 IEEE International Conference on.* IEEE, 2015, pp. 227–231.

[151] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on.* IEEE, 2007, pp. 1–8.

[152] B. Waldvogel, "Accelerating random forests on cpus and gpus for object-class image segmentation." Master's thesis, Universität Bonn, 2013.

[153] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* Springer, 2013, vol. 6.

[154] [Online]. Available: https://msdn.microsoft.com/en-us/library/dn799273.aspx

[155] [Online]. Available: http://kinect.github.io/tutorial/lab11/index.html

[156] (2016, 02) Curfil paramaters. [Online]. Available: https://github.com/deeplearningais/curfil/

Vitae

- Name: Ridwan Shamsullah Jalali
- Nationality: Saudi
- Date of Birth: 09/10/1989
- Email: *jalalirsh@hotmail.com*
- Permenant Address: Saudi Arabia, Madinah